TUGAS AKHIR TF 145565

PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRAKTIKUM SISTEM INSTRUMENTASI INDUSTRI

DYAH RENGGANIS PERMATA DEWI
NRP 10511500000045

Dosen Pembimbing
Detak Yan Pratama, ST., M,Sc.
NIP. 19840101 201212 1 002

Murry Raditya, S.T., M.T
NIP. 1988201711055

PROGRAM STUDI D3 TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TUGAS AKHIR – TF 145565

PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRAKTIKUM SISTEM INSTRUMENTASI INDUSTRI

Dyah Rengganis Permata Dewi
NRP. 10511500000045

Dosen Pembimbing

Detak Yan Pratama, S.T., M.Sc.
NIP. 19840101 201212 1 002

Murry Raditya, S.T., M.T.
NIP. 1988201711055

PROGRAM STUDI D3 TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

FINAL PROJECT – TF 145565

DESIGN OF MONITORING FLOW RATE SYSTEM ON PIPE LEAK TEST EQUIPMENT AS SUPPORT PRACTICAL WORK FOR INDUSTRIAL INSTRUMENTATION SYSTEM

Dyah Rengganis Permata Dewi
NRP. 10511500000045

Advisor Lecture

Detak Yan Pratama, ST., M.Sc.
NIP. 19840101 201212 1 002

Murry Raditya, S.T., M.T.
NIP. 1988201711055

STUDY PROGRAM OF D3 INSTRUMENTATION TECHNOLOGY
INSTRUMENTATION ENGINEERING DEPARTEMENT
FACULTY OF VOCATION
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2018

LEMBAR PENGESAHAN

PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRAKTIKUM PADA SISTEM INSTRUMENTASI INDUSTRI

TUGAS AKHIR

Oleh :

Dyah Rengganis Permata Dewi
NRP. 10511500000045

Surabaya, 20 Juli 2018
Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Detak Yan Pratama, ST., M.Sc.	**Murry Raditya, S.T., M.T.**
NIP. 19840101 201212 1 002	**NIP. 198820171 1055**

Kepala Departemen Teknik Instrumentasi ITS

Dr.Ir. Purwadi Agus Darwito, M.Sc.
NIP. 19620822 198803 1 001

LEMBAR PENGESAHAN

PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRAKTIKUM PADA SISTEM INSTRUMENTASI INDUSTRI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Ahli Madya
pada
Program Studi D3 Teknologi Instrumentasi
Departemen Teknik Instrumentasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Oleh :
Dyah Rengganis Permata Dewi
NRP. 10511500000045

Disetujui oleh Tim Penguji Tugas Akhir :

1. Detak Yan Pratama, S.T., M.Sc. ..... Pembimbing I
2. Murry Raditya, S.T., M.T. ..... Pembimbing II
3. Dwi Oktavianto W.N, S.T., M.T. ..... Penguji I
4. Ahmad Fauzan Adziimaa, S.T., M.Sc. ..... Penguji II

SURABAYA
JULI 2018

PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRAKTIKUM SISTEM INSTRUMENTASI INDUSTRI

Nama : Dyah Rengganis Permata Dewi
NRP : 10511500000045
Jurusan : D3 Teknologi Instrumentasi
Pembimbing : Detak Yan Pratama, S.T., M.Sc.
Murry Raditya, S.T., M.T.

Abstrak

Ketidakseragaman dimensi pipa tentu menghasilkan kinerja yang berbeda. Aliran bertekanan dalam pipa menuntut kekuatan pipa yang memadai. Sehingga diperlukan pengujian untuk mengetahui kekuatan pipa. Adapula alat uji kebocoran pipa yang dihasilkan dari suatu aliran fluida yang bertekanan. Untuk mengetahui laju aliran pada alat uji diperlukannya sistem monitoring. Sistem monitoring laju aliran pada Alat Uji Kebocoran Pipa mutlak diperlukan untuk memastikan keamanan, melindungi peralatan dan untuk melakukan pemantauan. Menggunakan sensor water flow dengan Hall-Effect yang akan diproses dengan ATmega128 untuk mengetahui debit yang mengalir per satuan menit. Hasil pembacaan laju aliran akan dikirim pada LCD, HMI dengan visual basic serta disimpan pada SD Card. Berdasarkan hasil data pengujian sensor diperoleh nilai standar dari kalibari sensor, yaitu sebesar 0.23 L/menit, error pembacaan sensor sebesar 1.82% dan akurasi pembacaan sebesar 98.17%. Hasil pembacaan sensor akan disimpan pada memori dengan kapasitas 8 GB dan dapat bertahan $\pm$4 tahun lamanya.

Kata kunci : Monitoring, Water Flow, HMI, Logger

DESIGN OF MONITORING FLOW RATE SYSTEM ON PIPE LEAK TEST EQUIPMENT AS SUPPORT PRACTICAL WORK FOR INDUSTRIAL INSTRUMENTATION SYSTEM

Name **: Dyah Rengganis Permata Dewi**
NRP **: 10511500000045**
Departement **: Diploma of Instrumentation Technology**
Advisor Lecturer **: Detak Yan Pratama, S.T., M.Sc.**
Murry Raditya, S.T., M.T.

Abstract

Pipe dimension varicosity certainly produce a different performance.. Power flow in the pipe demanding sufficient pipe strength. So testing is required to find out the strength of the pipe. Pipe leak test tools there that are generated from a pressurised fluid flow. To know the flow rate on the test tools required the existence of a system for monitoring. Flow monitoring systems in pipe leak test equipment are absolutely necessary to ensure safety, protect the equipment and for monitoring. Using water flowmeter with Hal-Effect sensors are processed through ATmega128 to determine the flowing discharge per minute unit. The results of the reading flow rate will be sent to LCD, HMI in visual basic dan loggering to SD Card. Based on testing result got the standard value from calibration sensor that equal to 0.23 L/min, error of flow sensor reading equal to 1.82% and accuracy value of sensor reading equal to 98.17%. The result of reading flow sensor will be stored on memory with a capacity of 8 GB an can last approximately ±4 years.

Keywords: Monitoring, Water Flow, HMI, Logger

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul **"PERANCANGAN SISTEM MONITORING LAJU ALIRAN PADA ALAT UJI KEBOCORAN PIPA SEBAGAI PENUNJANG PRATIKUM SISTEM INSTRUMENTASI INDUSTRI"** tepat pada waktunya.

Terselesaikannya laporan ini juga tak luput dari dukungan dan peran serta dari orangtua dan keluarga besar serta berbagai pihak. Untuk itulah dalam kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Dr. Ir.Purwadi Agus Darwito,M.Sc selaku Kepala Departmen D3 Teknik Instrumentasi ITS yang telah memberikan semangat dan motivasi kepada kami
2. Bapak Detak Yan Pratama, S.T., M.Sc. dan Murry Raditya, S.T., M.T. selaku dosen pembimbing yang telah meluangkan waktu, arahan dan saran yang membantu selama proses pengerjaan tugas akhir
3. Papa dan Mama tercinta yang telah memberikan segala dukungan baik moral maupun materil serta dukungan yang sangat luar biasa berupa doa.
4. Terimakasih kepada Teman Sekelompok Tugas Akhir (Shaski, Zima, Edhy, Dicky, Ayom) yang bersama-sama berjuang dalam pengerjaan tugas akhir ini hingga akhirnya dapat terselesaikan.
5. Bapak Saiful Mad selaku mekanik bengkel yang sudah sangat membantu dalam pembuatan alat.
6. Teman-teman D3 Teknik Instrumentasi angkatan 2015 dan F50 yang yang telah memberi dukungan.

7. Serta semua pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa penulisan laporan Tugas Akhir ini tidaklah sempurna. Oleh karena itu sangat diharapkan kritik dan saran yang membangun dari semua pihak sehingga mencapai sesuatu yang lebih baik lagi. Penulis juga berharap semoga laporan ini dapat menambah wawasan yang bermanfaat bagi pembacanya.

Surabaya, 20 Juli 2018

Penulis

Dyah Rengganis P.D.
NRP. 10511500000045

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Setiap hari kita selalu berhubungan dengan fluida tanpa kita sadari. Sebagai contoh kita dapat melihat instalasi perpipaan air pada rumah yang kita tempati. Pipa sendiri merupakan saluran tertutup yang biasanya berpenampang lingkaran yang digunakan untuk mengalirkan fluida dengan tampang aliran penuh. Fluida yang di alirkan melalui pipa bisa berupa zat cair maupun gas. Pipa air pada instalasi rumah tangga umumnya terbuat dari bahan plastik yang banyak digunakan, salah satu jenis pipa yang paling umum adalah pipa *Poly Vinyl Chloride* (PVC) atau yang biasa disebut paralon. Produsen pipa PVC menentukan dimensi dan komposisi pipa tentu mengacu pada standar yang ada. Ketidakseragaman dimensi pipa tentu membedakan karakteristiknya. Dimensi, kekuatan, harga, dan standar pemasangannya dapat berbeda untuk pipa yang dibeli dari masing-masing toko. Adanya pipa PVC dengan berbagai dimensi dan kekuatan menghasilkan kinerja yang berbeda pula. Aliran bertekanan di dalam pipa menuntut kekuatan pipa yang memadai. Tentunya perlu bukti kongkrit untuk mengetahui bahwa kekuatan pipa PVC mampu menahan tekanan aliran fluida sehingga diperlukannya suatu pengujian. Jenis pengujian yang dibuktikan untuk membuktikan hal tersebut adalah uji kekuatan pipa PVC.

Salah satu metode yang digunakan untuk melakukan uji ketahan pipa PVC adalah menggunakan tekanan hidrostatik. Metode ini bertujuan untuk menguji kekuatan dari pipa PVC terhadap tekanan hidrostatik[1]. Hal utama yang diperlukan dalam melakukan pengujian tersebut adalah sebuah fluida cair, yaitu air. Dalam mengalirkan fluida cair pada pipa yang sedang diuji tidak diketahui debit fluida yang mengalir sehingga diperlukannya suatu sistem monitoring untuk hal tersebut. Oleh karena itu dibuatlah

tugas akhir ini dengan sistem monitoring laju aliran pada alat uji kebocoran pipa sebagai penunjang praktikum sistem instrumentasi industri. Adanya monitoring laju aliran pada alat uji kebocoran pipa adalah mutlak adanya, hal tersebut nantinya akan membantu dalam memonitor kerja sistem secara berkala, menjaga keamanan dan melindungi peralatan serta membuat operator lebih mudah untuk mengerti kerja dari sistem tersebut. Data yang ditampilkan pun dapat disimpan dan berguna sebagai bahan analisa apabila terjadi kejanggalan dalam kerja sistem tersebut. Sehingga perlunya dirancang komponen sistem monitoring yang dapat menampilkan pemantauan kontrol, pengambilan dan penyimpanan data, agar kinerja pada sistem lebih efisien.

1.2 Rumusan Permasalahan

Berdasarkan latar belakang yang dijelaskan sebelumnya, maka maka permasalah yang diangkat pada Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara merancang alat sistem monitoring laju aliran pada plant Alat Uji Kebocoran Pipa (AUKP)?
2. Bagaimana cara merekam dan menampilkan hasil data pembacaan variabel laju aliran pada plant Alat Uji Kebocoran Pipa (AUKP)?

1.3 Tujuan

Adapun tujuan utama dari penelitian Tugas Akhir ini adalah sebagai berikut:

1. Dapat merancang sistem monitoring laju aliran pada plant Alat Uji Kebocoran Pipa (AUKP).
2. Dapat merekam dan menampilkan hasil data variabel laju aliran pada plant Alat Uji Kebocoran Pipa (AUKP).

1.4 Batasan Masalah

Adapun batasan masalah dari penelitian Tugas Akhir ini adalah sebagai berikut:

1. Variabel yang digunakan adalah laju aliran.
2. Kontroler menggunakan Atmega128.
3. Sensor menggunakan *water flow* dengan *Hall-Effect*.
4. *Logger* menggunakan *SD Card*.

1.5 Manfaat

Adapun manfaat yang dapat diperoleh dari tugas akhir ini yaitu sebagai berikut

1. Tugas akhir ini dapat digunakan sebagai simulai maupun alat pratikum pada sejumlah mata kuliah yang ada pada departemen.
2. Tugas akhir ini digunakan sebagai bahan pembelajaran dan pengembangan untuk alat yang lebih baik.

Halaman Sengaja di Kosongkan

BAB II
TINJAUAN PUSTAKA

2.1 Fluida

Fluida adalah zat yang berubah bentuk secara terus menerus di bawah penerapan tegangan geser (tangensial) tidak peduli seberapa kecil tegangan gesernya. Fluida atau zat cair (termasuk uap air dan gas) dibedakan dari benda padat karena kemampuannya untuk mengalir. Fluida lebih mudah mengalir karena ikatan molekul dalam fluida jauh lebih kecil dari ikatan molekul dalam zat padat, akibatnya fluida mempunyai hambatan yang relatif kecil pada perubahan bentuk karena gesekan. Zat padat mempertahankan suatu bentuk dan ukuran yang tetap, sekalipun suatu gaya yang besar diberikan pada zat padat tersebut, zat padat tidak mudah berubah bentuk maupun volumenya, sedangkan zat cair dan gas, zat cair tidak mempertahankan bentuk yang tetap, zat cair mengikuti bentuk wadahnya dan volumenya dapat diubah hanya jika diberikan padanya gaya yang sangat besar. Gas tidak mempunyai bentuk maupun volume yang tetap,gas akan berkembang mengisi seluruh wadah. Karena fase cair dan gas tidak mempertahankan suatu bentuk yang tetap, keduanya mempunyai kemampuan untuk mengalir. Dengan demikian kedua – duanya sering secara kolektif disebut sebagai fluida.

Untuk mengerti aliran fluida maka harus mengetahui beberapa sifat dasar fluida. Adapun sifat – sifat dasar fluida yaitu: kerapatan (*density*) ρ, (*specific gravity*) (*s.g*), tekanan (*pressure*) P, kekentalan (*viscosity*) μ.

1. Kerapatan (*Density*)

Kerapatan (*density*) ρ suatu zat adalah ukuran untuk konsentrasi zat tersebut dan dinyatakan dalam massa per satuan volume. Sifat ini ditentukan dengan cara menghitung perbandingan massa zat yang terkandung dalam suatu bagian tertentu terhadap volume bagian tersebut.

2. Viskositas

Viskositas adalah ukuran ketahanan sebuah fluida terhadap deformasi atau perubahan-perubahan bentuk. Viskositas zat cair cenderung menurun dengan seiring bertambahnya kenaikan temperatur, hal ini disebabkan gaya-gaya kohesi pada zat cair bila dipanaskan akan mengalami penurunan dengan semakin bertambahnya temperatur pada zat cair yang menyebabkan berturunnya viskositas dari zat cair tersebut[2].

2.2 Laju Aliran (*Flow*)

Debit atau laju aliran merupakan suatu koefesien yang menyatakan jumlah volume air yang mengalir dari suatu sumber dalam satuan waktu tertentu melalui suatu penampang air, sungai, saluran, pipa atau kran dan sebagainya, biasanya diukur dalam satuan liter/detik atau liter/menit. Dalam pengukuran fluida termasuk penentuan tekanan, kecepatan, debit, gradien kecepatan, turbulensi dan viskositas. Terdapat banyak cara melaksanakan pengukuran-pengukuran, misalnya langsung, tak langsung, gravimetrik, volumetrik, elektronik, elektromagnetik dan optik. Pengukuran debit secara langsung terdiri dari atas penentuan volume atau berat fluida yang melalui suatu penampang dalam suatu selang waktu tertentu. Metoda tak langsung bagi pengukuran debit memerlukan penentuan tinggi tekanan, perbedaan tekanan atau kecepatan dibeberapa dititik pada suatu penampang dan dengan besaran perhitungan debit. Metode pengukuran aliran yang paling teliti adalah penentuan gravimerik atau penentuan volumetrik dengan berat atau volume diukur atau penentuan dengan mempergunakan tangki yang dikalibrasikan untuk selang waktu yang diukur[3]. Pengukuran laju aliran merupakan salah satu hal yang sangat penting dalam proses plant boiler. Pengukuran debit air ditujukan untuk mengetahui kecepatan aliran pada satuan waktu. Untuk mengetahui debit maka harus mengetahui satuan

ukuran volume dan satuan ukuran waktu terlebih dahulu, karena debit berkaitan dengan satuan volume dan satuan waktu.

Untuk menentukan debit air menggunakan persamaan:

$$Q = \frac{V}{t} \quad (2.1)$$

dimana:

Q = debit (liter/s)
V = volume (liter)
t = waktu (s)

2.2.1 Klasifikasi Aliran

Secara garis besar jenis aliran dapat dibedakan atau dikelompokkan sebagai berikut:

1. Aliran Tunak (*steady)*

   Suatu aliran dimana kecepatannya tidak terpengaruh oleh perubahan waktu sehingga kecepatan konstan pada setiap titik (tidak mempunyai percepatan).

2. Aliran Tidak Tunak (*unsteady*)

   Suatu aliran dimana terjadi perubahan kecepatan terhadap waktu.

2.2.2 Tipe-tipe Aliran

Bilangan Reynolds merupakan bilangan yang tak berdimensi yang dapat membedakan suatu aliran dinamakan laminer, transisi dan turbulen.

$$R = \frac{V}{v} \quad (2.2)$$

dimana:

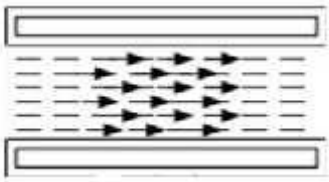
Re = bilangan reynolds
V = kecepatan fluida (m/s)

D = diameter dalam pipa (m)
v = viskositas dinamik fluida (kg/ms) atau (N.s/m²)

Adapun macam aliran fluida dapat diaktegorikan sebagai berikut[6]:

1. Aliran Laminar

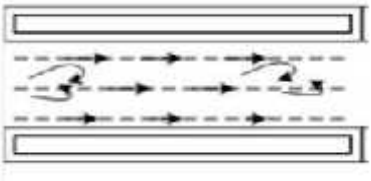
Aliran laminar didefinisikan sebagai aliran dengan fluida yang bergerak dalam lapisan–lapisan atau lamina–lamina dengan satu lapisan meluncur secara lancar. Aliran laminar ini mempunyai nilai bilangan Reynoldsnya kurang dari 2300 (Re < 2300).



Gambar 2.1 Aliran Laminer

2. Aliran Transisi

Aliran transisi merupakan aliran peralihan dari aliran laminer ke aliran turbulen. Keadaan peralihan ini tergantung pada viskositas fluida, kecepatan dan lain-lain yang menyangkut geometri aliran dimana nilai bilangan Reynoldsnya antara 2300 sampai dengan 4000 (2300<Re<4000) .

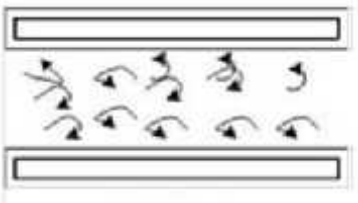


Gambar 2.2 Aliran Transisi

3. Aliran Turbulen

Aliran turbulen didefinisikan sebagai aliran yang dimana pergerakan dari partikel-partikel fluida sangat tidak menentu

karena mengalami percampuran serta putaran partikel antar lapisan, yang mengakibatkan saling tukar momentum dari satu bagian fluida ke bagian fluida yang lain dalam skala yang besar. Dimana nilai bilangan Renoldsnya lebih besar dari 4000 (Re>4000).



Gambar 2.3 Aliran Turbulen [2]

2.3 Pengukuran *Flow*

Pengukuran aliran fluida adalah sangat penting di dalam suatu industri proses seperti kilang minyak (*refinery*), pembangkit listrik (power plant) dan industri kimia (*petrochemical*). Pada industri proses seperti ini, memerlukan penentuan kuantitas dari suatu fluida (liquid, gas atau steam) yang mengalir melalui suatu titik pengukuran, baik didalam saluran yang tertutup (*pipe*) maupun saluran terbuka (*open channel*). Kuantitas yang ditentukan antara lain ; laju aliran volume (*volume flow rate*), laju aliran massa (*mass flow rate*), kecepatan aliran (*flow velocity*).

Instrumen untuk melakukan pengukuran kuantitas aliran fluida ini disebut *flowmeter*. Pengembangan *flowmeter* ini melalui tahapan yang luas mencakup pengembangan *flow* sensor, interaksi sensor dan fluida melalui penggunaan teknik komputasi (*computation techniques*), transduser dan hubungannya dengan unit pemprosesan sinyal (*signal processing units*), serta penilaian dari keseluruhan sistem di bawah kondisi ideal, kondisi gangguan (*disturbed*), kasar (*harsh*), kondisi berpotensi meledak (*explosive conditions*) serta pada lokasi laboratorium dan lapangan (*field*). Di

dalam pemilihan sensor *flow*, berikut kondisi-kondisi yang sangat berpengaruh dan harus diketahui untuk perhitungan, antara lain :

1. Ukuran pipa dimana laju aliran diukur
2. Daerah laju aliran
3. Karakteristik fluida :
   - Cairan, Gas, *Slurry*, dll.
   - Tekanan
   - Suhu
   - Viskositas
   - *Specific gravity*
   - Kompresibilitas
   - *Molecular weight* (*for gases and vapors*)
4. Pengaruh korosif (untuk membantu didalam pemilihan material)
5. Jenis aliran yang diukur termasuk aliran stabil atau aliran fluktuasi[4].

2.4 Sensor *Water Flow*

Pada rancang bangun *plant* Alat Uji Kebocoran Pipa (AUKP) ini untuk mengetahui laju aliran fluida dari *storage tank* menuju *pressure tank* digunakan sensor *water flow*.

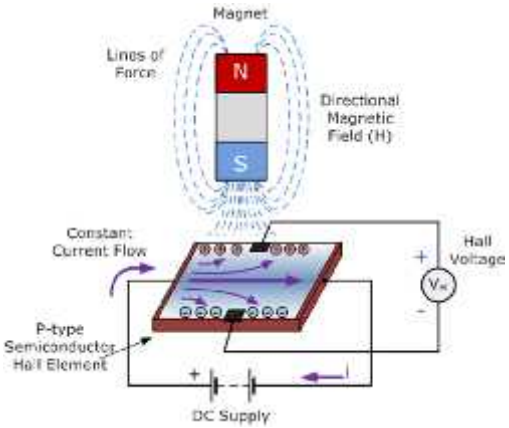
Sensor *water flow* merupakan sensor yang mendeteksi aliran air yang melewati sensor tersebut. Sensor ini terdiri dari tubuh katup plastik, rotor air, dan sensor *hall-effect*[5].

Gambar 2.4 Sensor *Water Flowmeter*

Air yang mengalir akan melewati katup dan akan membuat rotor magnet berputar dengan kecepatan tertentu sesuai dengan tingkat aliran yang mengalir. Medan magnet yang terdapat pada rotor akan memberikan efek pada sensor *hall-effect* dan itu akan menghasilkan sebuah sinyal pulsa yang berupa tegangan (Pulse Width Modulator). Sensor ini tidak akan menghasilkan tegangan apabila sensor belum dialiri air atau belum bekerja dan baru akan menghasilkan tegangan ketika sensor telah di aliri air.

Output dari pulsa tegangan memiliki tingkat tegangan yang sama dengan input dengan frekuensi laju aliran air. Sinyal tersebut dapat diolah menjadi data digital melalui pengendali atau mikrokontroler. Kelebihan sensor ini adalah hanya membutuhkan 1 sinyal (SIG) selain jalur 5V DC dan Ground. Berikut adalah persamaan untuk mengubah frekuensi menjadi debit, yaitu[6] :

$$F = 5.4\ x\ Q \tag{2.3}$$



Gambar 2.5 Prinsip Kerja Sensor *Hall-Effect*

Untuk *wiring* sensor *water flow* ini dengan ATmega terbilang cukup mudah. *Water flow* terdiri dari tiga pin, yaitu pin VCC yang

akan diberikan tegangan sebesar 5Volt sebagai sumber, kemudian pin *output* yang berupa sinyal pulsa dan yang terakhir adalah GND.

Berikut ini merupakan spesifikasi dari FS300A sensor *water flow* :

- Minimal tegangan operasional: DC 4.5V
- Maksimal arus operasional: 15mA (DC 5V)
- Rentang tegangan operasional: DC 5V~24V
- Rentang pembacaan debit : 1~60L/min
- Suhu operasional: ≤80℃
- Suhu cairan: ≤120℃
- Maksimal tekanan air: ≤2.0Mpa[7]

2.5 *Monitoring*

Monitoring didefinisikan sebagai siklus kegiatan yang mencakup pengumpulan, peninjauan ulang, pelaporan, dan tindakan atas informasi suatu proses yang sedang diimplementasikan. Umumnya, *monitoring* digunakan dalam *checking* antara kinerja dan target yang telah ditentukan. Untuk melakukan monitoring diperlukan alat yang didalamnya terdapat suatu sistem yang bertujuan untuk monitoring. *Monitoring* sendiri merupakan pengawasan pada suatu variabel atau sistem yang mana bertujuan untuk mengamati atau mengawasi keadaan sistem secara real time. Monitoring dilakukan untuk mendeteksi jika akan terjadi suatu kegagalan atau gangguan pada sistem sehingga dapat meminimalkan kegagalan atau gangguan tersebut. Selain berfungsi sebagai pengawas *monitoring* juga berfungsi untuk merekam (*record*) apa yang terjadi pada sistem yang tengah dimonitor dalam bentuk data tabel maupun grafik yang ditampilkan dalam bentuk display[8].

2.6 Mikrokontroler

Mikrokontroler adalah suatu chip berupa IC (*Integrated Circuit*) yang dapat menerima sinyal *input*, mengolahnya dan memberikan sinyal *output* sesuai dengan program yang diisikan ke dalamnya. Sinyal input mikrokontroler berasal dari sensor yang merupakan informasi dari lingkungan sedangkan sinyal *output* ditujukan kepada aktuator yang dapat memberikan efek ke lingkungan. Jadi secara sederhana mikrokontroler dapat diibaratkan sebagai otak dari suatu perangkat/produk yang mempu berinteraksi dengan lingkungan sekitarnya[9].

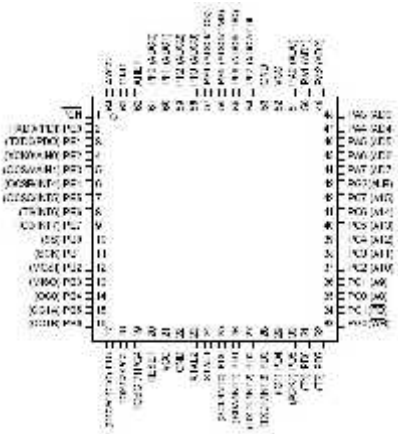
Mikrokontroler pada dasarnya adalah komputer dalam satu chip, yang didalamnya terdapat mikroprosesor, memori, jalur *Input/Output* (I/O) dan perangkat pelengkap lainnya. Kecepatan pengolahan data pada mikrokontroler lebih rendah jika dibandingkan dengan PC. Pada PC kecepatan mikroprosesor yang digunakan saat ini telah mencapai orde GHz, sedangkan kecepatan operasi mikrokontroler pada umumnya berkisar antara 1 – 16 MHz. Begitu juga kapasitas RAM dan ROM pada PC yang bisa mencapai orde Gbyte, dibandingkan dengan mikrokontroler yang hanya berkisar pada orde byte/Kbyte. Meskipun kecepatan pengolahan data dan kapasitas memori pada mikrokontroler jauh lebih kecil jika dibandingkan dengan komputer personal, namun kemampuan mikrokontroler sudah cukup untuk dapat digunakan pada banyak aplikasi terutama karena ukurannya yang kompak. Mikrokontroler sering digunakan pada sistem yang tidak terlalu kompleks dan tidak memerlukan kemampuan komputasi yang tinggi.

Mikrokontroler tersusun dalam satu chip dimana prosesor, memori, dan I/O terintegrasi menjadi satu kesatuan kontrol sistem sehingga mikrokontroler dapat dikatakan sebagai komputer mini yang dapat bekerja secara inovatif sesuai dengan kebutuhan sistem. Sistem *running* bersifat berdiri sendiri tanpa tergantung dengan komputer sedangkan parameter komputer hanya digunakan untuk download perintah instruksi atau program.

ATmega128 merupakan salah satu varian dari mikrokontroler AVR 8-bit. Beberapa fitur yang dimiliki adalah memiliki beberapa memori yang bersifat *non-volatile*, yaitu 128 Kb of *In-System Self-Programmable Flash program memory* (128 *Kbytes memory flash* untuk pemrograman), 4 Kb memori EEPROM, 4 Kb memori internal SRAM, *write/erase cycles* : 10.000 *flash*/ 100.000 EEPROM (program dalam mikrokontroler dapat diisi dan dihapus berulang kali sampai 10.000 kali untuk flash memori atau 100.000 kali untuk penyimpanan program/data di EEPROM).

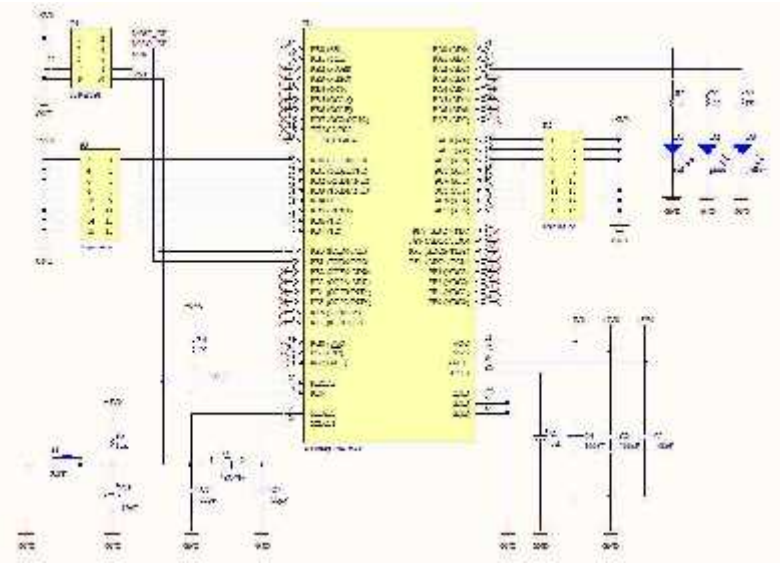
Gambar 2.6 ATMega128

Selain memori, fitur yang dimiliki oleh mikrokontroler atmega128 ini adalah pada perangkat *peripheral interface*, yaitu memiliki 2 buah 8-bit *timer/counter*, 2 buah *expand* 16-bit *timer/counter*, RTC (*Real Time Counter*) dengan *oscillator* yang terpisah, 2 buah 8-bit chanel PWM, 6 PWM chanel dengan resolusi pemrograman dari 2 sampai 16 bits, output compare modulator, 8 chanel 10-bit ADC, 2 buah TWI (*Two Wire Interface*), 2 buah serial USARTs, *master/slave* SPI *serial interface*, *Programmable Watchdog Timer* dengan *On-chip Oscillator*, *On-chip analog comparator*, dan memiliki 53 *programmable* I/O. Sedangkan untuk pengoperasiannya sendiri, Miktrokontroler ATmega128 dapat dioperasikan pada catuan 4.5 – 5.5 V untuk ATmega128 dengan *clock speed* 0–16 MHz.



Gambar 2.7 Pinout ATMega128

Sistem minimum merupakan suatu rangkaian minimalis yang dirancang supaya suatu mikrokontroler dapat berfungsi dan bekerja dengan semestinya. Sama seperti mikrokontroler lainnya, atmega128 juga membutuhkan sistem minimum. Namun sistem minimum pada Mikrokontroler ATmega128 memiliki beberapa perbedaan dibandingkan dengan sistem minimum mikrokontroler keluarga AVR yang lain. Perbedaan terletak pada konfigurasi pin pada ISP (*In System Programming*). Jika pada kebanyakan mikrokontroler jenis AVR konfigurasi pin untuk ISP adalah mosi-mosi, miso-miso, sck-sck, reset-reset, dan power supply, maka pada Mikrokontroler ATmega128 adalah mosi-RX0, miso-TX0, SCK-SCK, dan power supply[10].



Gambar 2.8 Minimum Sistem Atmega128

Desain sistem minimum tersebut merupakan rangkaian minimum yang terdiri dari beberapa led indikator dan 2 port I/O expansion, selain itu juga dilengkapi dengan rangkaian referensi clock, rangkaian reset, dan port pemrograman ISP. Pada rangkaian sistem minimum ini juga harus diperhatikan bahwa pin PEN harus pada kondisi pull up (pin PEN dihubungkan dengan catuan/vcc yang diberi tahanan). Selain itu juga perlu diperhatikan bahwa untuk konfigurasi programing mikrokontroler atmega 128 ini menggunakan ISP, pin MOSI downloader terhubung dengan pin RX0 mikrokontroler, sedangkan pin MOSI downloader terhubung dengan pin TX0 mikrokontroler, sedangkan pin SCK dan pin Reset downlaoder masing masing terhubung dengan pin SCK dan pin Reset mikrokontroler. Port-port I/O dan peripheral interface pada Mikrokontroler ATmega128 yang telah terhubung dengan sistem minimum dapat langsung dihubungkan ke perangkat-

perangkat atau komponen lainnya untuk diintegrasikan menjadi suatu sistem elektronika yang lebih kompleks[9].

2.7 LCD 20x4

Pada sistem monitoring, *display* yang digunakan pada plant yaitu LCD (*liquid Crystal Display*). LCD sendiri merupakan suatu perangkat elektronika yang telah terkonfigurasi dengan kristal cair dalam gelas plastik atau kaca sehingga mampu memberikan tampilan berupa titik, garis, simbol, huruf, angka ataupun gambar. LCD terbagi menjadi dua macam berdasarkan bentuk tampilannya, yaitu *Text*-LCD dan *Grapic*-LCD. Berupa huruf atau angka, sedangkan bentuk tampilan pada *Graphic*-LCD berupa titik, garis dan gambar. Dalam LCD setiap karakter ditampilkan dalam matriks 5x7 pixel. Pada gambar di bawah ini merupakan LCD 20 x 4 yang berguna untuk menampilkan pembacaan sensor pada plant yang telah di olah di mikrokontroler dan kemudian ditampilkan ke LCD untuk menjadi *interface* hasil pembacaan sensor[11].

Gambar 2.9 LCD 20 × 4 cm

Berikut adalah konfigurasi pin pada LCD 20x4 :

Tabel 2.1 Konfigurasi Pin LCD 20 × 4 cm.

PIN NO.	SYMBOL	FUNCTION
1	V_{SS}	Ground
2	V_{DD}	+3 V or +5 V
3	V_0	Contrast adjustment
4	RS H/L	register select signal
5	R/W H/L	read/write signal
6	E H/L	enable signal
7	DB0 H/L	data bus line
8	DB1 H/L	data bus line
9	DB2 H/L	data bus line
10	DB3 H/L	data bus line
11	DB4 H/L	data bus line
12	DB5 H/L	data bus line
13	DB6 H/L	data bus line
14	DB7 H/L	data bus line
15	A	Power supply for LED (4.2 V)
16	K	Power supply for B/L (0 V)

2.8 Komunikasi Data Serial

Komunikasi serial adalah komunikasi yang pengiriman datanya per-bit secara berurutan dan bergantian. Komunikasi ini mempunyai suatu kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya komunikasi serial merupakan komunikasi dimana pengiriman data dilakukan per bit sehingga lebih lambat dibandingkan komunikasi parallel, atau dengan kata lain komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu.

Pada dasarnya komunikasi serial adalah kasus khusus komunikasi paralel dengan nilai n = 1, atau dengan kata lain adalah suatu bentuk komunikasi paralel dengan jumlah kabel hanya satu dan hanya mengirimkan satu bit data secara simultan.Hal ini dapat disandingkan dengan komunikasi paralel yang sesungguhnya di mana n-bit data dikirimkan bersamaan, dengan nilai umumnya $8 \leq n \leq 128$.

Terdapat dua macam komunikasi serial, yaitu *asynchronous serial* dan *synchronous serial*. *Synchronous serial* adalah komunikasi dimana hanya ada satu pihak (pengirim atau penerima) yang menghasilkan *clock* dan mengirimkan *clock* tersebut bersama-sama dengan data. Contoh pengunaan *synchronous serial* terdapat pada transmisi data keyboard. *Asynchronous serial* adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan *clock* namun hanya data yang ditransmisikan, tanpa *clock*. Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi *clock* harus sama dan harus terdapat sinkronisasi. Setelah adanya sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi *clock* pengirim dan penerima akan membaca data sesuai dengan frekuensi *clock* penerima[12].

2.8.1 SPI

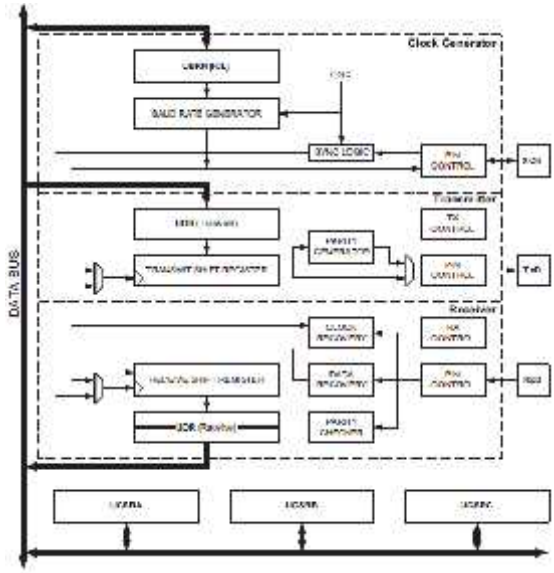
Serial Peripheral Interface atau SPI juga menyediakan komunikasi serial dua arah antara *receiver* dan *transmitter*. Dalam sistem SPI, *transmitter* dan *receiver* berbagi *common clock source* (SCK). Hal ini membutuhkan *line clock* tambahan tetapi memungkinkan untuk tingkat transmisi data yang lebih tinggi dibandingkan dengan USART. Sistem SPI memungkinkan pertukaran data yang cepat dan efisien antara mikrokontroler atau perangkat periferal. Ada banyak sistem eksternal kompatibel SPI yang tersedia untuk memperluas fitur mikrokontroler. Misalnya, LCD *(Liquid Crystal Display)* atau ADC (Analog to Digital

Converter) dapat ditambahkan ke mikrokontroler menggunakan sistem SPI.

Dalam Komunikasi SPI, dua atau lebih device yang terhubung yang salah satunya akan bertindak sebagai Master dan yang lainnya akan bertindak sebagai Slave. Master Device adalah device yang memulai sambungan dan melakukan control transmisi data. Saat kedua device sudah terkoneksi, master dapat meminta request data baik mengirim data ataupun menerima data. Seperti pada penjelasan di atas, hal ini dinamakan koneksi *full duplex*, yaitu master device dapat mengirim data dan juga slave device juga dapat mengirim data pada saat yang bersamaan

2.8.2 USART

USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) merupakan salah satu perangkat yang digunakan untuk melakukan komunikasi serial dari mikrokontroler ATMega 128. Serial USART menggunakan metode *full duplex* (dua arah) antara *receiver* dan *transmitter*



Gambar 2.10 Blok Diagram USART ATmega128

USART memiliki 2 pin (RxD dan TxD) untuk *Asynchronous* dan 3 bit TxD, RxD, xCK untuk *Synchronous*. Untuk mengatur komunikasi USART dilakukan melalui beberapa register yaitu :

1. UDR (*USART Data Register*) adalah register yang paling penting dalam komunikasi serial ini. Sebab data yang dikirim keluar harus ditempatkan pada register ini, sedang data yang diterima dari luar dapat dibaca pada register ini pula. Pada intinya register UDR digunakan sebagai *buffer* untuk menyimpan data, baik yang akan dikirim maupun yang akan diterima. Sejatinya UDR adalah terdiri dari 2 buah register terpisah, dengan alamat dan nama yang sama, yakni UDR. Saat kita menulis data pada UDR ini, maka sebenarnya kita menulis data pada UDR (*Write*) yang kemudian USART mem-frame dengan bit-bit frame dan segera akan segera mengirimkan data tersebut secara serial. Saat kita membaca UDR, sebenarnya adalah membaca UDR (*Read*). Data yang diterima secara serial akan disimpan dalam register tersebut, setelah hadirnya stop bit, maka USART akan membuang frame dan menyiapkan data pada UDR (*Read*) sehingga dapat segera di ambil.
2. UCSRA (*USART Control dan Status Register A*) adalah register yang penting. Sebegian besar adalah berisi status dari dari proses transfer komunikasi serial itu sendiri. Adapun penjelasan dari bit-bit tersebut adalah:
   - Bit 7 – RxC: *USART Receive Complete*
     Bit ini menjadi tinggi jika ada data yang masih belum diambil atau dibaca di dalam *buffer* penerima. Bit ini akan otomatis rendah setelah *buffer* penerima telah dibaca. Jika Unit Penerima tiba-tiba dimatikan setelah diaktifkan, maka isi dalam b*uffer* penerima akan langsung dibuang dan bit RxC ini akan langsung dibuat rendah. Bit ini juga bisa mengaktifkan instrupsi *Receive Complete interrupt*.

- Bit 6 – TxC: *USART Transmit Complete*
  Bit ini akan otomatis tinggi saat semua frame dalam *shift-register* pengiriman telah digeser semuanya keluar dan jika tidak ada data baru yang berarada dalam *buffer* pengiriman. Bit TxC ini akan otomatis rendah setelah *Transmit Complete interrupt* dijalankan.
- Bit 5 – UDRE: *USART Data Register Empty*
  Bit UDRE ini adalah untuk menjadikan tanda jika *buffer* pengiriman telah siap untuk diberikan data baru. Bit ini akan bernilai 1 (tinggi) , berarti saat itu boleh menulis UDR.

- Bit 4 – FE: *Frame Error*
  Bit ini otomatis menjadi tinggi jika saat menerima data, ternyata ada kesalahan dari *frame* yang diterima.
- Bit 3 – DOR: *Data OverRun*
  Bit ini akan menjadi tinggi saat kondisi *overrun* terjadi. Kondisi ini terjadi saat *buffer* penerima sudah penuh dan berisi 2 data karakter, dimana data karakter terakhir tidak bisa dipindahkan ke UDR-*read*, karena tidak kunjung dibaca oleh user.
- Bit 2 – PE: *Parity Error*
  Bit ini akan menjadi tinggi saat karakter yang sedang diterima ternyata memiliki format parity yang salah. Tentu saja hal ini terjadi jika bit parity checking diaktifkan.
- Bit 1 – U2X: *Double the USART Transmission Speed*
  Bit ini hanya berlaku untuk operasi tak sinkron (*asynchronous*). Jika bit ini kita tulis dengan 1 (tinggi) maka *baud rate* akan menjadi 2 kali lebih cepat. Hal itu terjadi karena pembagi *baud rate* yang biasanya membagi 16 kemudian menbagi menjadi dengan 8 saja. Tulis bit ini dengan 0 (rendah) untuk operasi sinkron (*synchronous*).

- Bit 0 – MPCM: *Multi-processor Communication Mode*
  Bit ini digunakan untuk mode komunikasi *Multi-Prosesor*. Saat bit PMCM ini dibuat menjadi tinggi maka setiap data yang diterima oleh unit penerima, namun tidak dilengkapi dengan informasi alamat, data yang benar, maka akan diabaikan. Bit ini hanya berguna untuk penerima, dan bukan untuk pengirim.

3. UCSRB (USART Control dan Status Register B) adalah register yang penting. Sebagian besar adalah berisi status dari dari proses transfer komunikasi serial itu sendiri. Adapun penjelasan dari bit-bit tersebut adalah:
   - Bit 7 – RxCIE: *Rx Complete Interrupt Enable*
     Menulis bit ini menjadi tinggi akan mengaktifkan instrupsi yang berkaitan dengan bit RxC. *USART Receive Complete interrupt* akan terjadi hanya jika bit RxCIE ini dan bit I (*Global Inteurpt*) milik register SREG adalah 1 dan ada tingginya bit RxC milik UCSRA.
   - Bit 6 – TxCIE: *Tx Complete Interrupt Enable*
     Menulis bit ini menjadi tinggi maka akan mengaktifkan interupsi pada bit TxC, yakni *USART Transmit Complete interrupt* yang akan menjalankan interupsi setiap frame dari data pengiriman selasai dikirim atau dengan kata lain terjadi interupsi setiap bit TxC menjadi tinggi. Interupsi hanya bisa terjadi jika sebelumnya bit TxCIE ini di set tinggi dan bit *Global Interupt* (I) milik register UCSRA juga di-set tinggi.
   - Bit 5 – UDRIE: *USART Data Register Empty Interrupt Enable*
     Menulis bit ini menjadi tinggi maka akan mengaktifkan interupsi pada bit UDRE, yakni *Data Register Empty interrupt* yang akan menjalankan interupsi saat data pada *buffer* pengiriman sudah kosong atau dengan kata lain

setiap bit UDRE menjadi tinggi. Interupsi hanya bisa terjadi jika sebelumnya bit UDRIE ini di set tinggi dan bit *Global Interupt* (I) milik register UCSRA juga di-set tinggi.

- Bit 4 – RxEN: *Receiver Enable*
  Agar unit penerima dari USART dapat bekerja, maka bit RxEN ini harus dibuat tinggi sebelumnya. Begitu dibuat tinggi, pin RxD akan diputus sebagai standar I/O dan dihubungkan dengan unit penerima USART ini. Namun jika tiba-tiba bit RxEN ini dibuat rendah kembali setelah tadinya tinggi, maka unit penerima USART akan segera menghentikan kerjanya dan membatalkan proses penerimaan data, serta membersihkan *buffer* penerimaan, termasuk juga bit FE, DOR, dan PE.
- Bit 3 – TxEN: *Transmitter Enable*
  Agar unit pengirim USART dapat bekerja, maka bit TxEN ini harus dibuat tinggi sebelumnya. Begitu dibuat tinggi, pin TxD akan diputus dari standar I/O dan dihubungkan dengan unit pengirim USART ini. Namun jika tiba-tiba bit TxEN ini dibuat rendah kembali setelah tadinya tinggi, maka unit pengirim USART masih harus menyelesaikan tugasnya yang terakhir, yakni mengirim data yang tersisa. Baru kemudian unit pengiriman USART akan berhenti dan mengembalikan port TxD menjadi standar I/O kembali.
- Bit 2 – UCSZ2: *Character Size*
  Bit ini adalah pasangan dari Bit UCSZ1 dan bit UCSZ0 milik register UCSRC, untuk menentukan jumlah data yang hendak ditransfer.
- Bit 1 – RxB8: *Receive Data Bit 8*
  Jika kita menggunakan format penerimaan data 9 bit maka bit yang terakhir yang diterima akan ditempatkan pada bit RxB8 ini. Pabrik menyarankan untuk membaca bit ini

terlebih dahulu sebelum membaca 8 bit data lainnya di UDR.

- Bit 0 – TxB8: *Transmit Data Bit 8*
  Jika kita menggunakan format pengiriman data 9 bit maka bit yang terakhir yang hendak dikirim ditempatkan pada bit TxB8 ini. Pabrik menyarankan untuk menulis bit ini terlebih dahulu sebelum menulis 8 bit data lainnya di UDR.

4. UCSRC (USART Control dan Status Register C) adalah register penting, untuk melakukan kontrol pada peralatan USART. Namuin jika tidak menggunakan fungsi USART, maka boleh mengabaikan register ini seperti dalam keadaan resetnya.
   - Bit 7 – URSEL: *Register Select*
     Saat akan menulis port $20 maka ada dua register yang akan akan diakses, yang ditentukan dari D7 dari data yang dituliskan. Jika D7 (MSB) adalah 1 atau datanya diatas $80 maka sedang menuliskan data pada UCSRC. Sedang jika data yang hendak dituliskan adalah dibawah $80, maka kita sedang menulis UBRRH.
   - Bit 6 – UMSEL: *USART Mode Select*
     Bit ini untuk memilih USART dijadikan mode tak sinkron (*Asynchronous*) atau sinkron (*Synchronous*).
   - Bit 5:4 – UPM1:0: *Parity Mode*
     Bit-bit ini adalah untuk menghidupkan pembangkit dan pemeriksa parity. Jika diaktifkan maka akan otomatis membangkitkan parity pada setiap data yang dikirimkan dan akan memeriksa validitas parity dari setiap data yang diterima. Pada Unit penerima kita akan mendapatkan parity yang diterima, kemudian parity tersebut akan dibandingkan dengan status dari UMP0. Jika ternyata tidak cocok, maka bendera bit PE (Parity Error) pada UCSRA akan diaktifkan.

- Bit 3 – USBS: *Stop Bit Select*
  Dengan membiarkan bit ini menjadi 0 maka frame akan dilengkapi dengan *stop-bit* selebar 1 bit. Sedang jika bit ini ditulis tinggi, maka bit stop menjadi 2 bit.
- Bit 2:1 – UCSZ1:0: *Character Size*
  Menentukan karakter dari data yang hendak dikirimkan dan diterima haruslah merujuk pada bit-bit ini. Yakni bit UCSZ1 dan bit UCSZ0 milik register ini. Ditambah dengan bit UCSZ2 pada register UCSRB.
- Bit 0 – UCPOL: *Clock Polarity*
  Bit ini hanya digunakan pada mode *synchronous*. Dalam mode ini akan direlasikan antara daya yang diterima dan data yang dikirm dan disinkronkan dengan status dari XCX (*synchronous clock*).

5. UBRR (USART Baud Rate Register) adalah register yang menentukan *baud rate* dari peralatan USART. Sebagaimana biasanya, *baud rate* antara AVR dan device target haruslah sama. Hal ini menjadi sangat penting agar tidak ada transfer data palsu yang isinya diluar kemauan.
   - Bit 15 – URSEL: *Register Select*
     Bit ini untuk memilih dalam mengakses alamat port $20. Seperti yang kita tahu bahwa alamat port ini dimiliki oleh 2 buah regsiter yang berbeda, yakni UBRRH dan UCSRC. Saat kita hendak menulis UBBRH maka pastikan data bit MSB atau D7 harus rendah.
   - Bit 14:12 – *Reserved Bits*
     Bit ini tidak digunakan saat ini. Oleh karena itu kita disarankan untuk tidak menulisnya dengan nilai 1 pada bit-bit yang tidak digunakan.

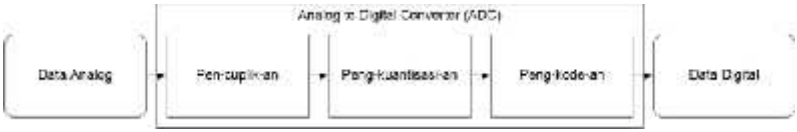
- Bit 11:0 – UBRR11:0: *USART Baud Rate Register*
  Ini adalah merupakan register 12 bit yang mana berisi dengan pengatur *baud rate* USART. Register ini adalah pasangan dari register UBBRH dan UBRRL, dimana UBBRL berisi 8 bit LSB dan sisanya dimiliki oleh UBRRH. Mengubah nilai dari register ini saat transmisi berlangsung, akan menyebabkan kesalahan transmisi, karena *baud rate* akan langsung berubah seiring dengan berubahnya nilai dari register ini[13].

2.9 Analog Digital

Sinyal analog adalah sinyal data dalam bentuk gelombang yang mempunyai nilai kontinyu (tidak terputus) dimana besarannya berubah terhadap waktu atau ruang, dan mempunyai semua nilai untuk setiap nilai waktu (dan atau setiap ruang), yang membawa informasi dengan mengubah karakteristik gelombang. Dalam instrumentasi biasanya besarnya sinyal analog adalah 4 – 20 mA, 0 – 20 mA. Sinyal digital adalah sinyal data dalam bentuk pulsa yang mengalami perubahan yang tiba-tiba dan mempunyai besaan 0 dan 1. Sinyal digital hanya memiliki dua keadaan yaitu 0 dan 1, sehingga tidak mudah dipengaruhi oleh noise. Dalam instrumentasi biasanya besarnya 0 dan 24V, 0 dan 5V. Dalam dunia instrumentasi analog input ini berasal berbagai macam transmitter ataupun analizer yang mengeluarkan sinyal 4–20 mA. Transmitter ataupun Analizer mengeluarkan sinyal 4–20mA adalah merupakan input bagi kontroler yang nantinya dapat dipakai sebagai indikasi ataupun set poin untuk alarm maupun interlock dalam suatu sistem.

ADC (*Analog to Digital Converter*) adalah sebuah rangkaian elektronika yang dapat mengubah besaran analog menjadi besaran digital. Pada setiap sensor yang berbasis mikrokontroler (sebagai pusat pengolah data) diperlukan adanya rangkaian ADC untuk mengubah sinyal yang diterima oleh sensor untuk menjadi besaran digital supaya sinyal tersebut bisa diterjemahkan atau dibaca

mikrokontroler. ADC memiliki 2 karakter prinsip, yaitu kecepatan sampling dan resolusi. Kecepatan sampling suatu ADC menyatakan seberapa sering sinyal analog dikonversikan ke bentuk sinyal digital pada selang waktu tertentu. Kecepatan sampling biasanya dinyatakan dalam sample per second. Prinsip kerja ADC adalah mengkonversi sinyal analog ke dalam bentuk besaran yang merupakan rasio perbandingan sinyal input dan tegangan referensi.



Gambar 2.11 Diagram Blok Proses ADC

DAC (*Digital to Analog Converter*) adalah perangkat atau rangkaian elektronika yang berfungsi untuk mengubah suatu isyarat digital (kode-kode biner) menjadi isyarat analog (tegangan analog) sesuai harga dari isyarat digital tersebut. DAC dapat dibangun menggunakan penguat penjumlah inverting dari sebuah operasional amplifier (Op-Amp) yang diberikan sinyal input berupa data logika digital (0 dan 1) .

2.10 Data *Logger*

Data *logger* (perekam data) adalah sebuah alat elektronik yang mencatat data dari waktu ke waktu baik yang terintegrasi dengan sensor dan instrumen. Atau secara singkat data logger adalah alat untuk melakukan data *logging*. Secara fisik data logger berukuran kecil. Perangkatnya dilengkapi dengan mikroprosesor dan memori internal yang digunakan untuk mencatat dan merekam data dan sensor. Beberapa jenis data logger biasanya dikoneksikan dengan komputer dan untuk mengaktifkannya digunakan sebuah *software*

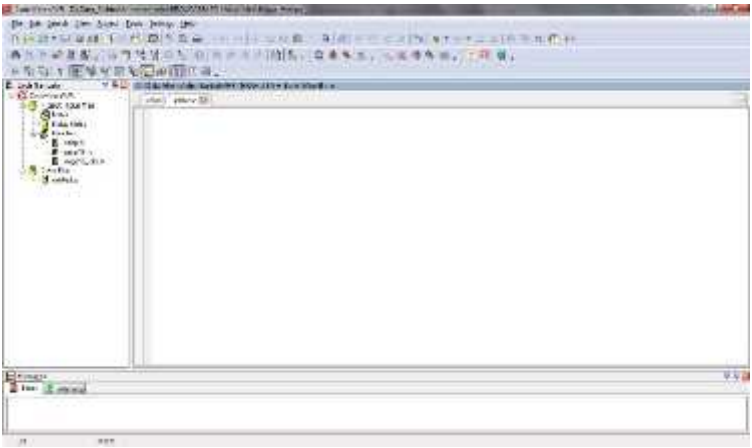
yang lebih simple. Pengamatan terhadap data yang terekam bisa dilakukan melalui komputer.

Data *logging* adalah proses otomatis pengumpulan dan perekaman data dari sensor untuk tujuan pengarsipan atau tujuan analisis. Sensor digunakan untuk mengkonversi besaran fisik menjadi sinyal listrik yang dapat diukur secara otomatis akhirnya dikirimkan ke komputer atau mikroprosesor untuk pengolahan. Berbagi macam sensor sekarang tersedia. Sebagai contoh, suhu, intesitas cahaya, tingkat suara, sudut rotasi, posisi, kelembaban relatif, pH, oksigen terlarut, pulsa (detak jantung), bernapas, kecepatan angin, dan gerak. Selain itu, banyak peralatan laboratorium dengan output listrik dapat digunakan bersama dengan konektor yang sesuai dengan data *logger*. Salah satu keuntungan menggunakan data logger adalah kemampuannya secara otomatis mengumpulkan data setiap 24 jam. Setelah diaktifkan, data *logger* digunakan dan ditinggalkan untuk mengukur dan merekam informasi selama periode pemantauan. Hal ini memungkinkan untuk mendapatkan gambaran yang komprehensif tentang kondisi lingkungan yang di pantau, contohnya seperti suhu udara dan kelembaban relatif[14].

2.11 Codevision AVR

CodeVision AVR merupakan sebuah software yang digunakan untuk memprogram mikrokontroler sekarang ini telah umum. Mulai dari penggunaan untuk kontrol sederhana sampai kontrol yang cukup kompleks, mikrokontroler dapat berfungsi jika telah diisi sebuah program, pengisian program ini dapat dilakukan menggunakan compiler yang selanjutnya diprogram ke dalam mikrokontroler menggunakan fasilitas yang sudah di sediakan oleh program tersebut. Salah satu compiler program yang umum digunakan sekarang ini adalah CodeVision AVR yang menggunakan bahasa pemrograman C.

CodeVision AVR mempunyai suatu keunggulan dari compiler lain, yaitu adanya *codewizard*, fasilitas ini memudahkan pengguna atau pemrogram dalam inisialisasi mikrokontroler yang akan digunakan[15].

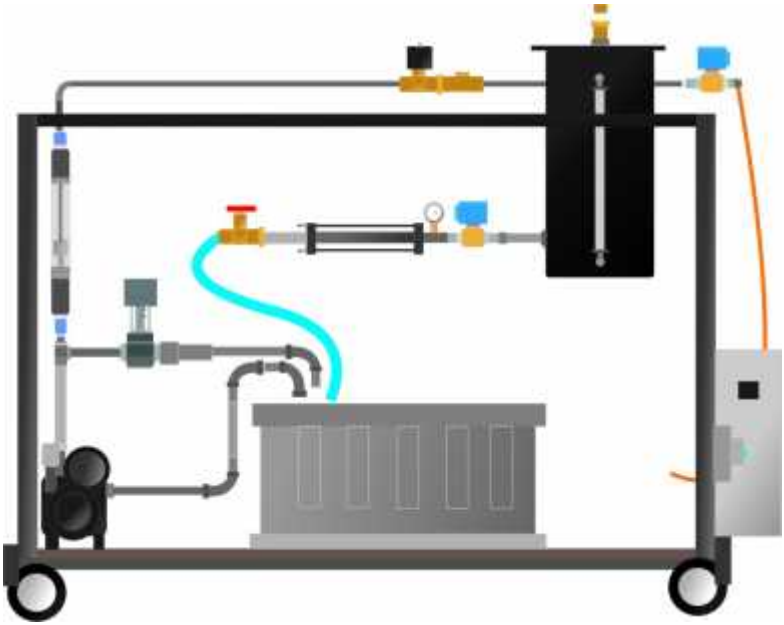


Gambar 2.12 Tampilan Codevision AVR

BAB III
PERANCANGAN DAN PEMBUATAN ALAT

3.1 Perancangan Alat

Berikut ini merupakan rancangan desain alat secara keseluruhan yang akan dijelaskan melalui diagram piping dan instrumen dibawah ini:



Gambar 3.1 Desain Alat Uji Kebocoran Pipa

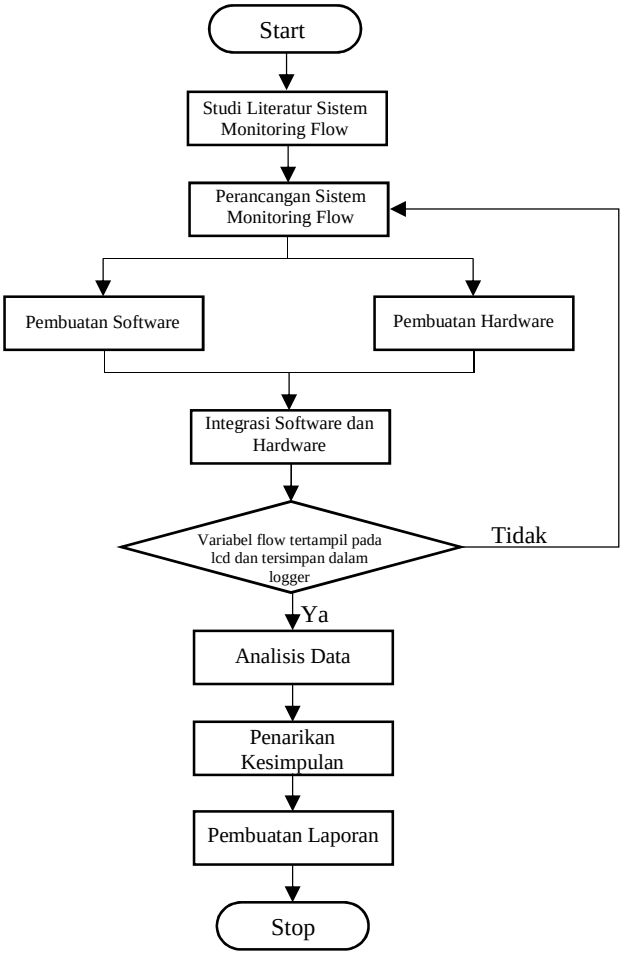
Dari gambar 3.1 diatas dapat dilihat proses pada alat uji tekanan pipa. Pompa *feedwater* digunakan untuk memompa air dari tangki penampung menuju *pressure tank* sebelum dilakukannya pengujian pipa. Pada pressure tank sejumlah air ditampung disana sebagai media untuk pengujian pipa uji nantinya. Pipa uji yang dibutuhkan sebagai pengujian hanyalah sepanjang 25 cm atau tiga

kali diameter pipa uji ditambah dua kali panjang kopeling sudah mewakilkan panjang pipa sebenarnya. Setelah ketinggian air yang diperlukan sudah terpenuhi maka seluruh posisi aktuator (motor stepper) dalam keadaan tertutup terkecuali aktuator pada bagian kompresor yang akan menyuplai udara bertekanan pada *pressure tank*. Guna penyuplaian udara bertekanan pada *pressure tank* adalah untuk memperoleh tekanan uji yang sesuai dengan tekanan kerja pipa uji. Setelah sejumlah udara bertekanan dimasukkan dalam *pressure tank* maka aktuator pada kompresor berubah menjadi dalam posisi tertutup menyegel udara bertekanan serta sejumlah air dalam *pressure tank*. Kemudian aktuator yang akan menyalurkan sejumlah air pada pipa uji dibuka sehingga air dengan tekanan yang telah dibuat tersebut menumbuk pipa uji. Membutuhkan durasa minimal sekitar satu jam ataupun seribu jam dalam pengujian pipa uji tersebut. Jika dalam kurun waktu sekian tidak terjadi kebocoran maupun kerusakaan pada pipa uji maka dapat dinyatakan pipa uji tersebut sesuai dengan tekanan kerja yang tertera dalam datasheet. Setelah pengujian pipa selesai *manual valve* sebagai penyegel pipa uji yang dekat dengan tangki penyimpanan dibuka supaya air yang tadinya digunakan sebagai pengujian dikembalika pada tangki penyimpanan.

Perancangan Sistem Monitoring Flow berada setelah pompa dengan penempatan sensor setelah aktuator, hal ini digunakan untuk mengetahui seberapa laju aliran yang keluar dari pompa serta menjaga laju aliran untuk mengisi *pressure tank.*

3.2 Diagram Alir (*Flowchart*)

Pada tugas akhir kali ini terdapat beberapa tahapan yang dilakukan untuk membuat sistem *monitoring flow* pada alat uji kebocoran pipa, sebagai berikut :



Gambar 3.2 Flowchart Perancangan Monitoring Flow

3.3 Pemilihan Komponen

Berikut ini komponen yang digunakan dalam sistem monitoring *flow* pada alat uji kebocoran pipa:

1. ATmega128
2. Sensor Water Flowmeter

3. Rotameter
4. Kabel
5. Jumper
6. LCD 20x4
7. SD Card
8. Modul *Openlog*
9. Kabel mini USB
10. Kabel *downloader*
11. Adaptor 12V

3.4 Indentifikasi Sistem Monitoring

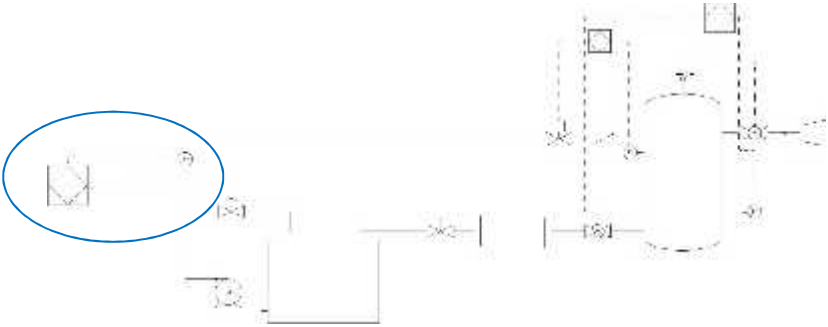
Pada identifikasi meliputi mencari dan mempelajari bahan pustaka maupun konsep – konsep yang berkaitan dengan segala permasalahan mengenai perancangan alat sistem monitoring laju aliran pada AUKP sebanyak-banyaknya, seperti Sensor Water Flow dengan Hall-Effect, LCD 20 × 4 cm, Mikrokontroler ATMega 128, dan lain lain.

3.5 Permodelan *Hardware* dan *Software*

Pada pemodelan ini dilakukan perancangan *hardware* dan *software* terlebih dahulu untuk sistem monitoring laju aliran pada AUKP. Diharapkan dengan dibuatnya perancangan tersebut supaya lebih memudahkan dalam hal pebuatan piranti alat nantinya sebelum menyentuh ranah pembuatan alat, seperti pirantik mekanik dan interface monitoring.

Gambar 3.3 Realisasi Pemasangan Sensor Water Flow

3.6 Perancangan dan Pembuatan *Hardware* dan *Software*

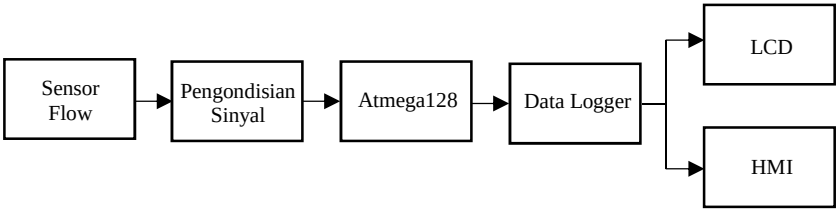


Gambar 3.4 P&ID Alat Uji Kebocoran Pipa

Perancangan dan pembuatan Piping and Instrumentation Diagram digunakan sebagai alat bantu berupa skema untuk menerangkan konsep desain dari sistem monitoring, meliputi: jalur perpipaan, peralatan yang diperlukan, serta sistem kontrol dari

proses yang berjalan. Pada diagram ini semua peralatan proses dan sistem instrumentasi digambarkan dalam bentuk simbol-simbol standar “Instrument Society of America” yang biasa disebut ISA Standart. Dapat dilihat pada gambar 3.4 yang mana merupakan desain rancang bangun AUKP

Dilakukan perancangan sistem monitoring laju aliran pada alat uji kebocoran pipa. Dimana pembuatan sistem monitoring dilakukan dengan membuat *hardware* serta *software*.

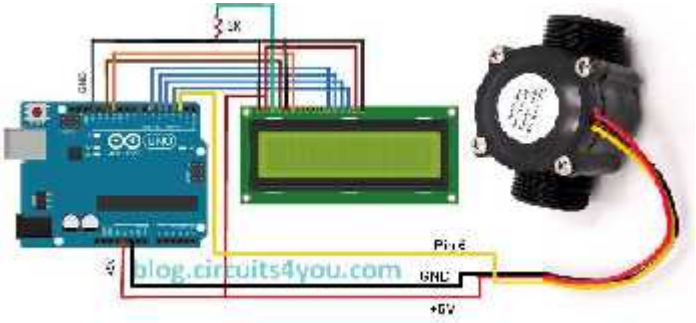


Gambar 3.5 Diagram Blok Sistem Monitoring Flow

Pada Gambar 3.5 dapat dilihat, ketika sensor mendeteksi adanya fluida yang mengalir maka sensor akan mengirimkan sinyal berupa pulsa yang kemudian masuk ke pengkondisian sinyal dan diubah menjadi data digital agar dapat dikirim menuju mikrokontroler. Mikrokontroler berfugsi memproses data untuk ditampilkan pada layar LCD dan HMI pada pc, selain itu mikrokontroler juga berfugsi mengirim data ke data logger shield untuk proses pencatatan data. Dimana pada data logger shield terpasang memory SD Card yang berfungsi sebagai media penyimpan hasil pencatatan data masukan laju aliran fluida.

1. Perancangan Sensor Water Flowmeter
   Sensor flow pada plant dipasang secara vertikal dengan pemasangan sesudah pompa. Pemasangan water flow sensor dipasang secara vertikal dengan ukuran sensor ¾ inch. *Water flowmeter* sensor ini memiliki 3 kabel yaitu

kabel merah yang disambungkan ke VCC, kuning ke output, dan hitam ke ground. Output dari sensor diletakkan pada PORT D bit 6 pada mikrokontroller atmega 128.



Gambar 3.6 Wiring Sensor Water Flow

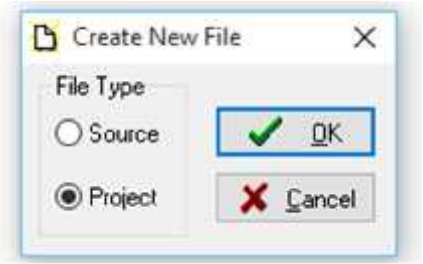
2. Perancangan Software

Perancangan software pada tahap ini merupakan siklus penggabungan untuk rangkaian pemrosesan sinyal, yaitu mikrokontroler ATmega128. Untuk membuat program pada ATMega128, dibutuhkan software CodeVisionAVR untuk memrogram mikrokontroler. Pada program ini, akan diberikan koding untuk mengolah sinyal masukan, dengan langkah – langkah sebagai berikut.

- Persiapkan alat dan bahan yang dibutuhkan (Sensor Water Flow, Laptop, ATmega128, Downloader ISP, Jumper, LCD, Modul Openlog, RTC).
- Pastikan software CodevisionAVR telah terinstal pada laptop yang akan digunakan.
- Buka software CodevisionAVR.

Gambar 3.7 Buka Software CodevisionAVR

- Ketika software berhasil dibuka, pilih toolbar *File* >> *New*, kemudian pada *file type* pilih "*project*" dan klik Ok.

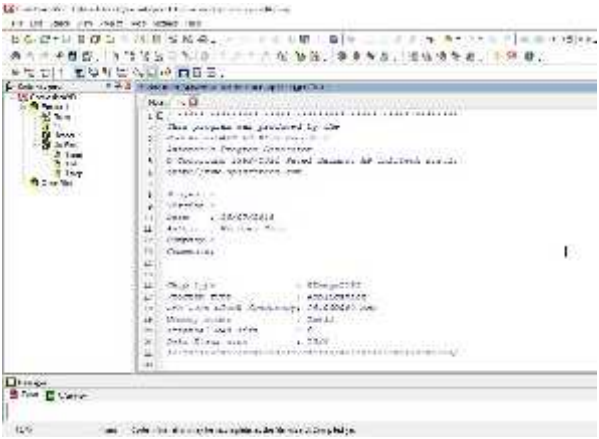


Gambar 3.8 Jendela *Create New File*

- Berikutnya akan muncul jendela pengaturan fitur-fitur/konfigurasi project dalam mikrokontroler AVR, seperti Ports, Chip, USART, I2C, Analog Comparator, dan sebagainya. Atur fitur-fitur yang diperlukan untuk merancang sistem monitoring laju aliran.

Gambar 3.9 Jendela Fitur AVR

- Pilih toolbar *Generate* untuk menyimpan project yang telah dibuat.
- Setelah itu akan muncul program yang telah diatur seperti gambar dibawah ini.



Gambar 3.10 Jendela Awal Program

- Mulailah untuk memprogram. Setelah selesai, *compile* program untuk mengetahui error yang ada.

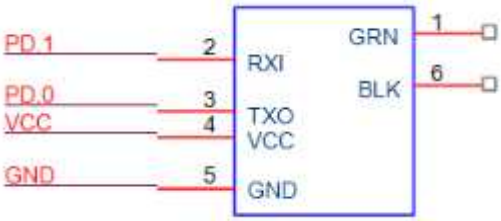
- Bila tidak terdapat error pada program kemudian upload program pada ATmega128 dengan downloader ISP.

Tabel 3.1 Alokasi Penggunaan Pin ATmega

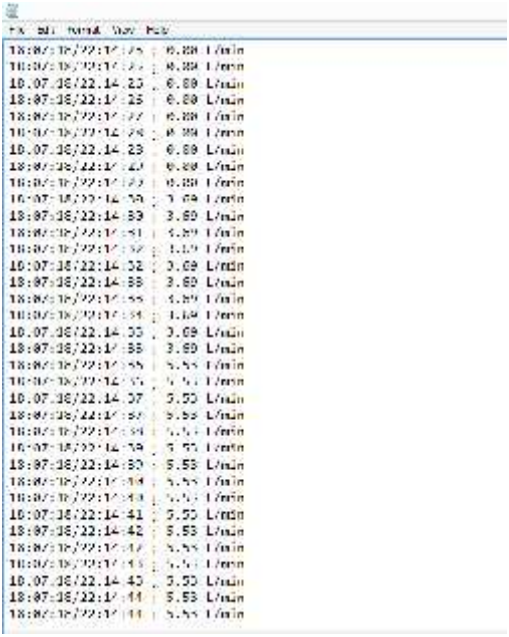
No.	Port	Fungsi
1.	Port C	LCD
2.	Port D0	SCL (TWI RTC)
3.	Port D1	SDA (TWI RTC)
4.	Port D2	RX USART1
5.	Port D3	TX USART1
6.	Port D6	Data Sensor
7.	Port mini usb	USART0

3. Perancangan Data Logger

   Untuk keperluan data logging maka digunakan Openlog data logger. Modul ini mempunyai 6 buah pin yaitu pin RX, TX, VCC, GND, GRN, dan BLK. Namun untuk penggunaan pada Atmega 128 hanya 4 buah pin yang dipakai. Komunikasi dengan modul ini menggunakan komunikasi serial USART. Untuk itu pin RX pada modul ini disambungkan dengan pin TX (PORT D bit 3) mikrokontroler dan pin TX pada modul disambungkan dengan pin RX (PORT D bit 2) pada mikrokontroler. Dan yang paling utama, pin GND harus disambungkan pada GND yang sama yang dipakai mikrokontroler guna mendukung komunikasi serial. Berikut ini adalah skematik dari konfigurasi modul Openlog data logger:



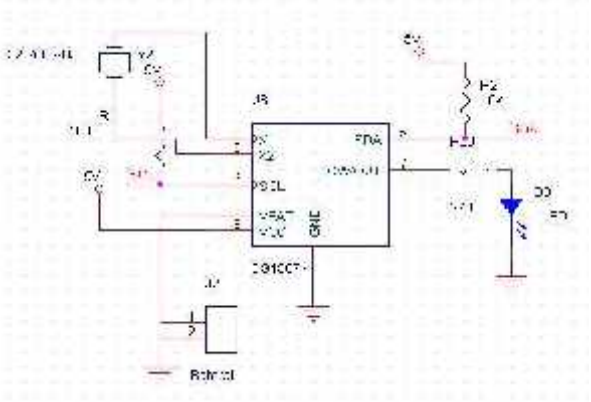
Gambar 3.11 Skematik Openlog Datalogger



Gambar 3.12 *File* Logger dalam SD Card

4. Perancangan *Real Time Clock* DS1307
   Pada saat sensor melakukan akuisisi data diperlukan data waktu dan tanggal guna mendukung proses *logging* data. Untuk kebutuhan tersebut, maka dirancanglah sebuah *real time clock* dengan menggunakan DS1307. DS1307 membutuhkan beberapa komponen pendukung yaitu

crystal, kapasitor, *battery backup* dan *pull up resistor* guna mendukung kinerjanya.

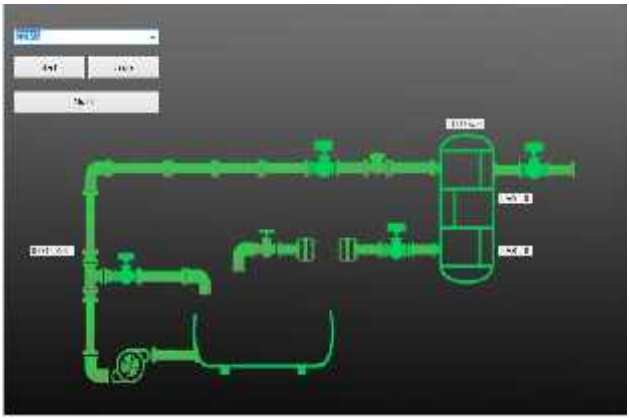


Gambar 3.13 Rangkaian RTC DS1307

DS1307 menyediakan pin battery backup untuk dihubungkan pada baterai lithium 3V atau sumber energi lain sehingga ketika supply daya utama mati, battery backup mengambil alih supply energi pada RTC sehingga DS1307 tetap berjalan.

5. Perancangan Display

   Pada perancangan tampilan untuk sistem monitoring laju aliran terbagi menjadi dua, yaitu LCD dan HMI. Modul LCD (Liquid Crystal Display) yang digunakan dalam tugas akhir ini adalah modul LCD yang dapat menampilkan 20x4 karakter, yaitu 20 karakter untuk kolom dan 4 karakter untuk baris. Kaki-kaki dari rangkaian LCD dihubungkan pada Port C mikrokontroler. Kemudian ada pula HMI (Human Machine Interface) yang akan menampilkan perubahan variable laju aliran melalui PC. Dimana data dari mikrokontroler dikirim pada PC menggunakan kabel mini usb.



Gambar 3.14 Desain HMI

3.7 Integrasi *Hardware* dan *Software*

Sistem monitoring pada alat ini akan menggunakan Atmega sebagai mikrokontrollernya dan menggunakan beberapa *interface* dalam menampilkan hasil monitoringnya yaitu LCD dan HMI *(Human Machine Interface)* serta untuk penyimpanan data jangka panjang akan menggunakan openlog dengan menggunakan SD Card. Dimana *listing program* yang telah selesai akan di integrasikan antara *hardware* dengan *software*. Kemudian *listing program* tersebut di upload pada ATmega128 menggunakan software khazamah. Penyesuaian interface antar software dan hardware dapat dilihat pada LCD ataupun dan komunikasi serial yang ditampilkan pada software HMI serta dilakukan pengecekan data yang tersimpan pada SD Card.

3.8 Pengujian Sistem

Pengujian sistem monitoring laju aliran ini dilakukan untuk memastikan integrasi sistem monitoring variabel proses pada alat dapat berjalan dengan seharusnya dan dapat menghasilkan data yang valid. Pengujian ini berfungsi sebagai penentu apakah hasil yang diharapkan sesuai atau tidak. Adapun tolok ukur untuk mengerahui keberhasilan dari pengujian sistem, antara lain:

- Dapat menampilan data variable proses pada LCD
- Dapat menampilkan dalam HMI dan mengamati perubahan yang terjadi
- Dapat menyimpan data dalam logger dengan media SD Card

3.9 Kalibrasi Sensor

Kalibrasi sensor dengan membandingkan sensor *water flow* dengan validator, yaitu rotameter. Dalam melakukan pengujian alat dan kalibrasi sensor, adapun beberapa tahapan perhitungan. Berikut beberapa persamaan yang digunakan :

1. Karakteristik statik
   - Sensitivitas

     $$\text{Sensitivitas} = \frac{\Delta O}{\Delta I} \quad (3.1)$$

   - Non-linieritas

     $$(N(I)) = O(I) - (KI+a) \quad (3.2)$$

     Non – linieritas maksimum per unit

     $$\frac{N}{O \quad -O} x\, 1 \quad \% \quad (3.3)$$

   - a (*zero bias*) $= O_{min} - KI_{min}$ (3.4)
   - Histerisis :

     $$H(I) = O(I)_{I\uparrow} - O(I)_{I\downarrow}, \; \hat{H} = H(I)_{max}$$

     % maksimum histerisis =

     $$\frac{\hat{H}}{O \quad -O} x\, 100\% \quad (3.5)$$

   - Akurasi

     $$A = 1 \left| \frac{Y \; -X}{Y} \right| x\, 100\% \quad (3.6)$$

   Dengan :

   Yn = Pembacaan Standar

   Xn = Pembacaan Alat

   - Kesalahan (*Error*)

     $$e = 1 - A \quad (3.7)$$

2. Kalibrasi Sensor *Water Flow*
   - Nilai Ketidakpastian tipe A :
     Standar deviasi :

$$\sigma = \frac{\sqrt{\Sigma(y_i - \bar{y})^2}}{n-1} \quad (3.8)$$

$$U_a = \frac{\sigma}{\sqrt{n}} \quad (3.9)$$

$$U_a = \sqrt{\frac{S}{n-2}}. \quad (3.10)$$

     Dimana :
     SSR *(Sum Square Residual)* = ΣSR(*Square Residual*)
     SR = R^2 (*Residu*)
   - Yi (Nilai koreksi) =
     Pemb. standar (ti) – Pemb. alat (xi) (3.11)

$$Y = a + (b\ x\ t_i) \quad (3.12)$$

$$a = \bar{y}_i + (b\ x\ \bar{t}_i) \quad (3.13)$$

$$b = \frac{n.\Sigma t_i y_i - \Sigma y_i.\Sigma t_i}{n.\ \Sigma t_i^2 - (\Sigma t_i)^2} \quad (3.14)$$

     Dengan :
     t_i = Pembacaan standar
     y = Nilai koreksi
     n = Jumlah data
   - Nilai ketidakpastian tipe B
     Pada ketidakpastian tipe B ini terdapat 2 parameter ketidakpastia, yaitu ketidakpastian Resolusi (U_{B1}) dan ketidakpastian alat standar (U_{B2}). Berikut ini adalah perhitungan ketidakpastian tipe B :

$$\mathbf{U_{B1}} = \frac{\frac{1}{2}x\,R}{\sqrt{3}} \quad (3.15)$$

$$\mathbf{U_{B2}} = \frac{a}{k} \quad (3.16)$$

   - Nilai ketidakpastian kombinasi U_c :

$$\mathbf{U_c} = \sqrt{U_{AI}{}^2 + U_{A2}{}^2 + U_{B1}{}^2 + U_{B2}{}^2} \quad (3.17)$$

Dengan kondisi V atau derajat kebebasan dari kedua tipe ketidakpastian, sebagai berikut :

V_a = n-1, sehingga :

$V_b = \frac{1}{2}(\frac{1}{R})^2 = \frac{1}{2}(\frac{1}{1})^2$ = 50 (berdasarkan table T)

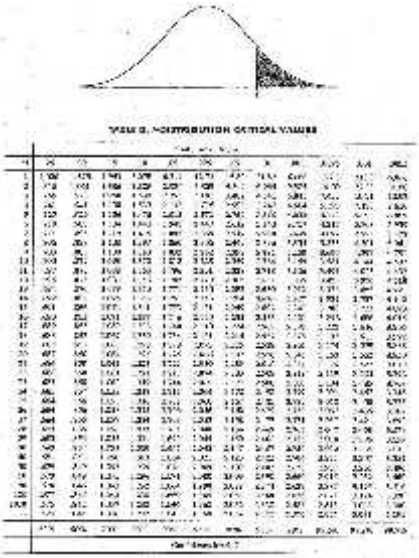
Dengan nilai Veff (Nilai derajat kebebasan effektif) sebagai berikut :

$$V = \frac{(U_c)^4}{\sum^{(U_i)^4}/_{V_i}} \quad (3.18)$$

Oleh karena itu, hasil nilai ketidakpastian diperluas sebesar:

$$U_e = k\ x\ U_c \quad (3.19)$$

Untuk mencari nilai k, maka melihat table t student sesuai dengan *confidence level* 95%. Tabel T *student* dapat dilihat pada gambar 3.6



Gambar 3.15 Tabel T Student

3.10 Analisa Data dan Penarikan Kesimpulan

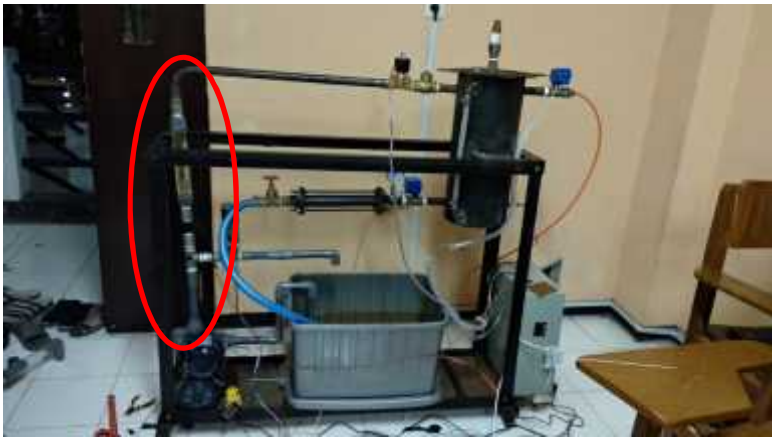
Tahap terkahir terdapat analisis data yang dilakukan bertujuan untuk mengetahui kinerja dari setiap piranti apakah sudah sesuai dengan perancangan *hardware* dan *software* dari hasil pengukuran, dan apabila tidak sesuai maka akan di cek ulang pada sistem integrasi dan uji kalibrasi sensor. Kemudian, dilakukan penarikkan kesimpulan dari semua elemen yang mempengaruhi data tersebut.

Halaman Sengaja di Kosongkan

BAB IV
HASIL DATA DAN PEMBAHASAN

4.1 Pengujian Alat

Berikut ini merupakan hasil perancangan sistem monitoring flow alat uji kebocoran pipa.



Gambar 4.1 Realisasi Alat Uji Kebocoran Pipa

Pada gambar 4.1 merupakan gambaran keseluruhan dari dari plant Alat Uji Kebocoran Pipa. Dimana nantinya plant tersebut dapat dipergunakan untuk melakukan uji kebocoran pipa plastik (PVC). Untuk sistem monitoring flow pada Alat Uji Kebocoran Pipa terletak sesudah pompa dengan posisi vertikal. Dimana peletakan sensor flow berada diantara rotameter dan konektor *three way* pipa.

Gambar 4.2 Sensor *Water Flow*

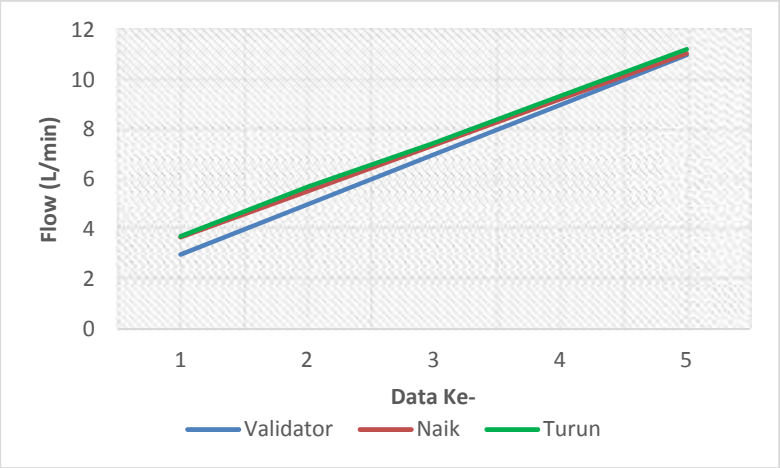
4.1.1 Pengujian Sensor *Flow*

Pada pembacaan sensor *flow* dilakukan pengambilan data sebanyak 5 kali dengan 5 variasi laju aliran yang telat ditetapkan. Dari pembacaan tersebut dapat dilihat besarnya nilai debit yang mengalir dari sensor *water flowmeter*. Berikut hasil data pembacaan sensor waterflow meter pada tabel 4.1 :

Tabel 4.1 Data Pembacaan Sensor *Water Flow*

Pembacaan Standar (L/min)	Pembacaan Alat		Rata-rata Pembacaan (L/min)	Koreksi (L/min)
	Naik (L/min)	Turun (L/min)		
3	3.69	3.73	3.71	-0.71
5	5.53	5.71	5.62	-0.62
7	7.38	7.47	7.425	-0.425
9	9.22	9.35	9.285	-0.285
11	11.07	11.23	11.15	-0.15
Jumlah			37.19	-2.19
Rata-rata			7.44	-0.44

Berdasarkan tabel 4.1 diatas dapat dilihat hasil debit dari pembacaan sensor *water flowmeter* yang dibandingkan dengan validator, yaitu rotameter. Sehingga diperoleh grafik pembacaan data naik dan data turun dari sensor water flow pada gambar 4.3, sebagai berikut:



Gambar 4.3 Grafik Histerisis Laju Aliran

4.1.2 Karakteristik Statik Sensor *Flow*

Karakteristik statik merupakan sifat dari sebuah alat ukur yang tidak bergantung terhadap waktu. Berikut merupakan data perhitungan untuk mengetahui nilai karakteristik suatu alat pada tabel 4.2 :

Tabel 4.2 Karakteristik Sensor

Pembacaan Standar (L/min)	Pembacaan Alat		Rata-rata Pembacaan (L/min)	Koreksi (L/min)	H(I)	H(I)/X
	Naik (L/min)	Turun (L/min)				
3	3.69	3.73	3.71	-0.71	0.04	0.013
5	5.53	5.71	5.62	-0.62	0.18	0.036
7	7.38	7.47	7.425	-0.425	0.09	0.013
9	9.22	9.35	9.285	-0.285	0.13	0.014
11	11.07	11.23	11.15	-0.15	0.16	0.015
Jumlah						0.091
Rata-rata						0.018

Berdasarkan persamaan 3.1 sampai dengan 3.7 pada bab 3, dihasikan nilai karakteristik statik sensor dari *water flowmeter* sebagaimana berikut :

a. Range : 3 - 11 L/min
b. Span : 8
c. Resolusi : 0,01
   Yang mana diperoleh nilai resolusi sebesar 0.01 berdasarkan pembacaan alat yang mana pembacaan dari sensor *water flow* dengan dua angka dibelakang koma.
d. Sensitivitas : 0.93
   Diperoleh nilai sensitivitas sensor sebesar 0.93 menandakan bahwa sensor water flow memiliki tingkat sensitivitas yang tinggi dalam mendeteksi laju aliran.
e. Non – Linieritas : 0.92
f. Histerisis : 2,44%

g. Akurasi : 98.17.%
   Diperoleh nilai akurasi pembacaan sensor sebesar 98,17% menandakan bahwa nilai pembacaan tersebut dapat dikatakan akurat.
h. *Error* : 1,82%
   Diperoleh nilai *error* sebesar 1,82% menandakan bahwa *error* pembacaan sensor *water flow* terbilang kecil.

4.1.3 Perhitungan Ketidakpastian Alat

Perhitungan ketidakpastian pembacaan alat dilakukan untuk mengetahui keadaan sensor *water flowmeter* yang digunakan masih bagus untuk dipakai pada sistem. Sehingga perlu dilakukan analisa dengan metode statistik (tipe A) dan selain metode statistik (tipe B)

Tipe A ditandai dengan adanya data pengukuran, selanjutnya dari data tersebut diperoleh nilai rata-rata dan standar deviasi. Terdapat 2 macam analisa tipe A yaitu UA1 dan UA2. UA1 merupakan ketidakpastian hasil pengukuran, sedangkan UA2 merupakan ketidakpastian regresi. Analisa Tipe B dibagi menjadi 2 bagian, yaitu ketidkpastian resolusi (UB1) dan ketidakpastian alat standar (UB2).

Dari semua sumber ketidakpastian tersebut dikombinasikan untuk memberikan gambaran menyeluruh ketidakpastian. Ketidakpastian gabungan biasa disebut dengan UC.

Berdasarkan persamaan kalibrasi yang terdapat pada persamaan 3.8 sampai 3.19 pada bab 3, maka didapatkan nilai-nilai ketidakpastian sebagaimana berikut :

a. U_{A1} : 0.103351343
b. U_{A2} : 0.381138186
c. U_{B1} : 0.002886751
d. U_{B2} : 0
e. U_{C} : 0.394912839

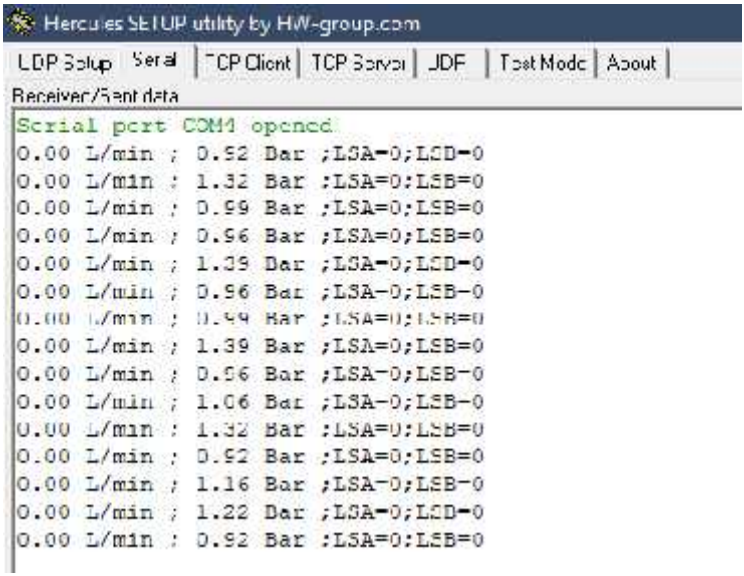
4.1.4 Pengujian *Hardware* Pada Sistem Monitoring

Pengujian hardware adalah untuk memastikan ATMega128 dengan open logger, RTC dan LCD sudah terpasang dengan benar. Dimana ATmega128 yang terpasang sebagai mikrokontroler dan terdapat wiring yang menuju mikrokontroler mulai dari sensor maupun aktuator. Tampilan dari data yang akan diterima oleh mikrokontroller dengan komunikasi serial USART akan menuju HMI (*Human Machine Interface*) pada PC serta akan tersimpan pada media penyimpanan SD CARD.

4.1.5 Pengujian Serial Data

Data dari pembacaan sensor yang telah diolah pada mikrokontroler kemudian dikirimkan menggunakan komunikasi serial USART. Pada ATmega128, pengiriman data menuju PC menggunakan USART0. Dimana pada pengiriman data menuju terminal tersebut menggunakan metode *polling*, yaitu metode pengendalian I/O melalui program. Semua pengalihan data dari dan ke alat I/O diselengarakan oleh program. Mikrokontroler mengirim dan meminta data sepenuhnya dibawah kendali program. Pengalihan data dapat dilaksanakan baik melalui mekanisme jabat tangan maupun tanpa jabat tangan. Dalam mekanisme jabat tangan isyarat diperiksa secara terus menerus. Program terus menerus berputar lewat sejumlah pengetesan untuk menentukan apakah masukan atau keluaran dapat diselenggarakan pelayanannya atau tidak. Yang mana kemudian data pada mikrokontroler akan langsung dikirim bila mikrokontroler dan PC telah terhubung dengan media penghubung kabel mini usb.

Pengujian yang dimaksudkan untuk mengetahui apakah data yang dikirimkan melalui komunikasi USART sudah sesuai dengan apa yang dirancang. Pengujian dilakukan dengan membuka software terminal bernama *Hercules Utility.*

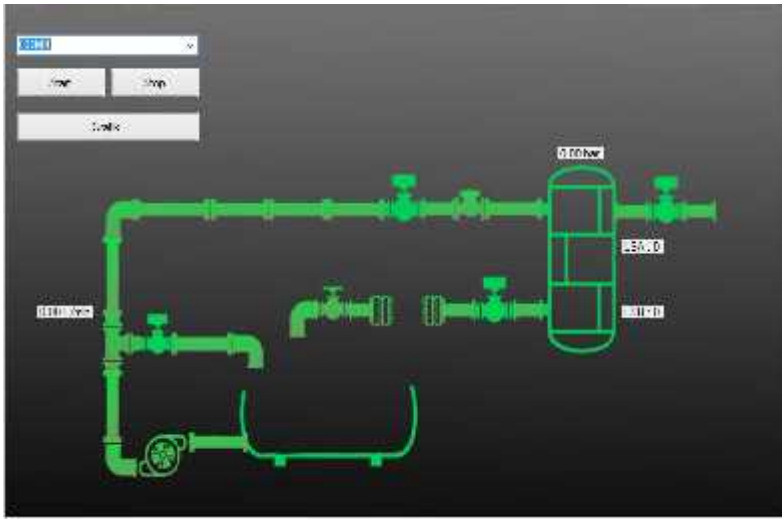


Gambar 4.4 Pembacaan Data Pada Terminal

Pada pengiriman data melalui terminal terdapat sekat berupa tanda " ; " yang nantinya berfungsi sebagai pembeda antara data masing-masing varible proses yang ada.

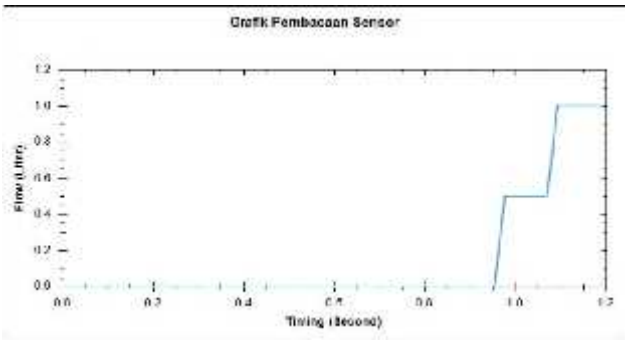
4.1.6 Pengujian *Display* Sistem *Monitoring Flow*

Pada sistem *monitoring flow* pada Alat Uji Kebocoran Pipa, data yang terbaca dapat ditampilkan pada LCD dan juga HMI di PC. Sehingga display dari monitoring flow sendiri tak hanya dapat dilihat melalui LCD yang terdapat pada panel box, namun juga tersambung pada HMI pada PC. Berikut tampilan *Human Machine Interface* (HMI) yang telah dibuat pada Visual Studio:



Gambar 4.5 Tampilan HMI pada PC

Pada gambar 4.5 merupakan tampilan utama HMI pada Alat Uji Kebocoran Pipa yang nantinya dapat menampilkan nilai debit dari pembacaan sensor. Untuk menjalankan HMI tersebut pastikan kabel mini usb telah tesambung pada port PC dan terkoneksi dengan ATmega. Setelah semua sambungan telah terkoneksi, tekan tombol start dan HMI akan melakukan pembacaan sesuai dengan keadaan sensor saat itu.



Gambar 4.6 Tampilan Grafik HMI pada PC

Pada gambar 4.6 merupakan bentuk tampilan dari pembacaan grafik ketika HMI dijalankan. Sehingga tak hanya menampilkan pembacaan laju aliran namun dapat memperlihatkan bentuk grafik dari pembacaan laju aliran itu sendiri.

Selain ditampilkan pada HMI di PC, data pembacaan sensor juga ditampilkan pada LCD karakter yang terletak pada panel box. Berikut tampilan LCD karakter yang berisi pembacaan data :

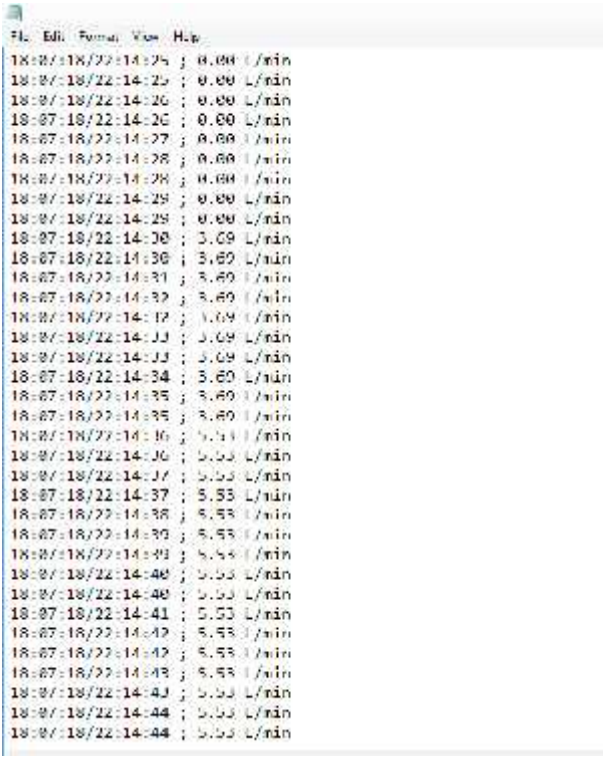
Gambar 4.7 Tampilan Pembacaan Sensor Pada LCD

Pada gambar 4.7 menampilkan data hasil pembacaan varibale proses yang ada pada LCD. Pada LCD karakter tersebut tersebut menampilkan 4 data variable proses, yaitu data sensor level switch, flow, data sensor tekanan, data sensor laju aliran serta keadaan motor stepper. Dimana untuk data laju aliransendiri diwakilkan dengan inisial huruf F pada LCD.

4.1.7 Pengujian *Openlog Data logger*

Penyimpanan data menggunakan *openlog datalogger* yang dapat berhasil disimpan pada format .txt. Pada modul openlog ini, data yang dikirim pada modul menggunakan komunikasi data serial berupa USART. Dimana modul tersebut menggunakan pin TX yang akan disambungkan pada RX mikrokontroler dan pin RX modul akan disambungkan pada pin TX mikrokontroler. Untuk pengiriman data logger menggunakan

ATmega128 dilakukan dengan USART1 yang mana berbeda dengan pengiriman data pada PC. Untuk protokol pengiriman data logger tak jauh berbeda dengan pengiriman data PC, yaitu menggunakan mode *polling*, dimana mode tersebut akan secara otomatis mengirim data pada logger bila modul openlog telah terhubung dengan mikrokontroler dan mikrokontroler memperoleh *supply* untuk menghidupkan ATmega



Gambar 4.8 Penyimpanan Data

Sistem monitoring laju aliran pada Alat Uji Kebocoran Pipa ini menggunakan modul *openlog data logger* sebagai penyimpanan data dengan SD Card yang berkapasitas 8GB. Pencatatan data

sensor *water flow* yang dilakukan selama satu menit lamanya membutuhkan ruang penyimpanan pada SD Card rata-rata sebesar 4 KB = 0.004 MB. Sehingga pencatatan data *water flow* dalam kurun waktu 1 hari akan membutuhkan kapasitas memori sebesar 5.09 MB. SD Card dengan kapasitas 8 GB memiliki nilai kapasitas maksimal yang dapat digunakan sebesar 7380 MB, Sehingga jumlah pencatatan yang dapat dilakukan dengan menggunakan SD Card yang bekapasitas 8 GB adalah sebagai berikut :

$$\text{Lama Waktu} = \frac{K \quad S \; C}{U \quad fi \; p \; h} \tag{4.1}$$

Dengan persamaan 4.1 maka penggunaan memori dapat digunakan selama 1449,90 hari atau kurang lebih selama 4 tahun lamanya.

4.2 Pembahasan

Pada pengambilan data *monitoring flow* diperoleh melalui pembacaan sensor *water flowmeter*. Sensor diletakkan sebelum memasuki *pressure vessel* dan sesudah pompa, hal ini dilakukan untuk mengetahui laju aliran yang keluar dari pompa supaya air dapat menjangkau inlet *pressure vessel*. Cara kerja dari alat ukur *flow* ini adalah masukan dari sensor yang berupa aliran air dapat menggerakkan turbin, sehingga turbin atau yang ada di dalam tubuh sensor bergerak dan menyebabkan terjadinya medan magnet karena adanya *hall effect*. Sedangkan keluaran dari sensor berupa sinyal pulsa, yang mana akan diolah oleh elemen pemrosesan sinyal yang berupa mikrokontroler Atmega128. *Listing program* disusun atau dikoding melalui software Codevision AVR, yang mana berfungsi untuk menampilkan data pengukuran dari sensor. Di dalam listing program CVAVR berisi rumus konversi yang dapat mengubah sinyal pulsa menjadi laju aliran dengan satuan liter per menit. Berdasarkan kalibrasi atau

pengujian sensor yang telah dilakukan dan diperoleh data pembacaan sensor *water flow* pada tabel 4.2 kemudian melalui tabel tersebut dilakukan perhitungan karakteristik statik sensor *water flowmeter* sehingga diperoleh nilai penyimpangan (standard deviation) dari kalibrasi sensor yakni sebesar 0,23 L/menit, error dari sensor *water flowmeter* sebesar 1.82%, nilai akurasi pembacaan sensor sebesar 98.17%. Dimana hasil dari perhitungan karakteristik statik membuktikan bahwa nilai error sensor sesuai dengan datasheet yang ada. Kemudian perhitungan ketidakpastian sensor guna mengetahui keadaan sensor apakah masih dalam keadaan bagus dalam memberikan nilai pembacaan. Hingga diperoleh nilai U_{A1} sebesar 0.1034, U_{A2} sebesar 0.3811, U_{B1} sebesar 0.0029, U_{B2} sebesar 0 dan U_C sebesar 0.395. Sistem monitoring flow ini pula juga memiliki tampilan pembacaan data sensor yang dapat dilihat melalui LCD karakter serta tampilan HMI pada PC ataupun melalui data logger yang tersimpan pada SD Card. Namun sebelum itu pastikan telah mengunduh program pada ATmega yang ada serta memastikan pemasangan pin pada sensor, karna kesalahan pemasangan dapat berpengaruh pada pembacaan sensor itu sendiri.

BAB V
PENUTUP

5.1 Kesimpulan

Adapun kesimpulan yang diperoleh ada tugas akhir ini, sebagai berikut :

1. Telah dirancang sistem monitoring laju aliran pada sistem Alat Uji Kebocoran Pipa, sehingga mempermudah untuk pemantauan dan melindungi keamanan peralatan.
2. Sensor *water flow* yang digunakan termasuk baik pada sistem karena nilai pembacaan error yang bernilai kecil, yaitu sebesar 1,82% dan akurasi pembacaan sensor yang terbilang baik pula dikarenakan memiliki nilai akurasi pembacaan sebesar 98.17%.

5.2 Saran

Adapun saran untuk penelitian selanjutnya, antara lain:

1. Dalam melakukan *wiring* disarankan untuk membuat modul pasti pada beberapa piranti agar tidak terlalu banyak jumper yang terpasang pada mikrokontroler.
2. Dalam perancangan sistem monitoring laju aliran dapat ditambahkan sistem IoT (*Internet of Thing*) yang mana dapat membuat pengguna dapat mengkases plant dari tempat manapun.

DAFTAR PUSTAKA

[1] http://www.indonesiainvestments.com/id/bisnis/komoditas/minyak-bumi/item267? [Diakses Juni 2017]

[2] https://academia.edu/8738082/Catatan_Mekanika_Fluida_ Disiapkan_oleh_Ridwan_Konsep_Dasar [Diakses Maret 2018]

[3] http://repository.usu.ac.id/bitstream/handle/123456789/47877/Chapter%20II.pdf [Diakses Maret 2018]

[4] Pertamina. 2007. Dasar Instrumentasi dan Proses Kontrol.

[5] http://repository.usu.ac.id/bitstream/123456789/33933/3/Chapter%20II.pdf [Diakses April 2018].

[6] https://teknikelektronika.com/pengertian-sensor-efek-hall-hall-effect-sensor-prinsip-kerja-efek-hall/ [Diakses Mei 2018

[7] Wiki. Datasheet G3/4 Water Flow Sensor.

[8] Mercy Corps. 2010. Design, monitoring, and evaluation guidebook.

[9] http://eprints.polsri.ac.id/2035/3/BAB%20II.pdf [Diakses April 2018]

[10] Atmel. 2011. Datasheet System ATmega128. San Jose, USA.

[11] Nurcahyo. (2012). Aplikasi dan Teknik Pemrograman Mikrokontroler AVR Atmel CV Andi Offset.

[12] http://arifzakariya.blog.ugm.ac.id/2012/01/09/komunikasi-serial-mikrokontroler/ [Diakses April 2018].

[13] http://izalhidayat.student.telkomuniversity.ac.id/register-usart-pada-mikrokontroler-atmega8535/ [Diakses Mei 2018]

[14] http://www.loggerindo.com/tahukah-anda-apa-itu-data-logger-29 [Diakses April 2018]

[15] Mukromin, Radian Indra. 2017. Laporan Tugas Akhir, Jurusan Teknik Instrumentasi ITS.

LAMPIRAN A
(*DATA SHEET SYSTEM ATMEGA128*)

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128Kbytes of In-System Self-programmable Flash program memory
  - 4Kbytes EEPROM
  - 4Kbytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Up to 64Kbytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels
    - 2 Differential Channels with Programmable Gain of 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V ATmega128L
  - 4.5 - 5.5V ATmega128
- Speed Grades
  - 0 - 8MHz ATmega128L
  - 0 - 16MHz ATmega128

8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash

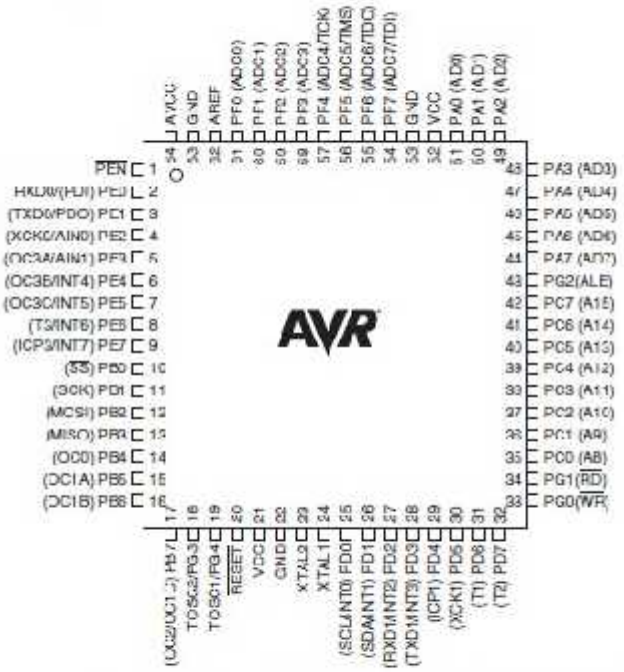
ATmega128
ATmega128L

Summary

Rev. 2467K-AVR-08/11

Pin Configurations

Figure 1. Pinout ATmega128



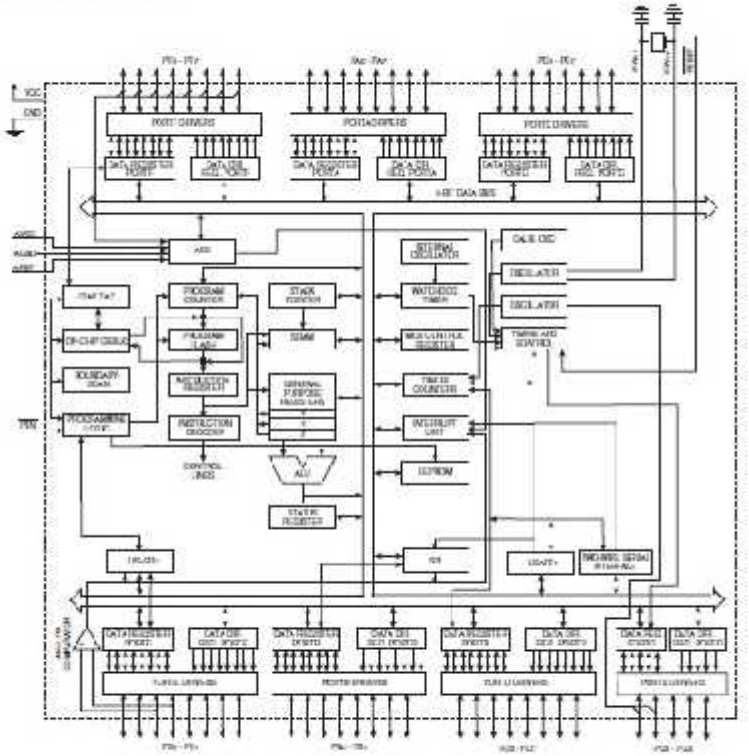
Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

Overview

The Atmel® AVR® ATmega128 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega128 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega128 provides the following features: 128Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 4Kbytes EEPROM, 4Kbytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8 bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega128 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega128 device is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

ATmega103 and ATmega128 Compatibility

The ATmega128 is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O locations reserved in the AVR instruction set. To ensure backward compatibility with the ATmega103, all I/O locations present in ATmega103 have the same location in ATmega128. Most additional I/O locations are allocated in an Extended I/O space starting from $60 to $FF, i.e., in the ATmega103 internal RAM space. These locations can be reached by using LD/LDS/LDD and ST/STS/STD instructions only, not by using IN and OUT instructions. The relocation of the internal RAM space may still be a problem for ATmega103 users. Also, the increased number of interrupt vectors might be a problem if the code uses absolute addresses. To solve these problems, an ATmega103 compatibility mode can be selected by programming the fuse M103C. In this mode, none of the functions in the Extended I/O space are in use, so the internal RAM is located as in ATmega103. Also, the Extended Interrupt vectors are removed.

The ATmega128 is 100% pin compatible with ATmega103, and can replace the ATmega103 on current Printed Circuit Boards. The application note "Replacing ATmega103 by ATmega128" describes what the user should be aware of replacing the ATmega103 by an ATmega128.

ATmega103 Compatibility Mode

By programming the M103C fuse, the Atmel® ATmega128 will be compatible with the ATmega103 regards to RAM, I/O pins and interrupt vectors as described above. However, some new features in ATmega128 are not available in this compatibility mode, these features are listed below:

- One USART instead of two, Asynchronous mode only. Only the eight least significant bits of the Baud Rate Register is available
- One 16 bits Timer/Counter with two compare registers instead of two 16-bit Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port C is output only.
- Port G serves alternate functions only (not a general I/O port).
- Port F serves as digital input only in addition to analog input to the ADC.
- Boot Loader capabilities is not supported.
- It is not possible to adjust the frequency of the internal calibrated RC Oscillator.
- The External Memory Interface can not release any Address pins for general I/O, neither configure different wait-states to different External Memory Address sections.

In addition, there are some other minor differences to make it more compatible to ATmega103:

- Only EXTRF and PORF exists in MCUCSR.
- Timed sequence not required for Watchdog Time-out change.
- External interrupt pins 7 - 0 serve as level interrupt only.
- USART has no FIFO buffer, so data overrun comes earlier.

Unused I/O bits in ATmega103 should be written to 0 to ensure same operation in ATmega128.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega128 as listed on page 72.

Port B (PB7..PB0) Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega128 as listed on page 73.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the Atmel® AVR® ATmega128 as listed on page 76. In ATmega103 compatibility mode, Port C is output only, and the port C pins are not tri-stated when a reset condition becomes active.

Note: The ATmega128 is by default shipped in ATmega103 compatibility mode. Thus, if the parts external programmer interface (by any future PCB, FORTE withdrawal out-lighted power up and will the ATmega103 compatibility mode is disabled.

Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega128 as listed on page 77.

Port E (PE7..PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega128 as listed on page 80.

Port F (PF7..PF0)

Port F serves as the analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a Reset occurs.

The TDO pin is tri-stated unless TAP states that shift out data are entered.

Port F also serves the functions of the JTAG interface.

In ATmega103 compatibility mode, Port F is an input Port only.

Port G (PG4..PG0)

Port G is a 5-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features.

The port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

	In ATmega103 compatibility mode, these pins only serves as strobe signals to the external memory as well as input to the 32kHz Oscillator, and the pins are initialized to PG0 = 1, PG1 = 1, and PG2 = 0 asynchronously when a reset condition becomes active, even if the clock is not running. PG3 and PG4 are oscillator pins.
RESET	Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 19 on page 58. Shorter pulses are not guaranteed to generate a reset.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.
AREF	AREF is the analog reference pin for the A/D Converter.
PEN	PEN is a programming enable pin for the SPI Serial Programming mode, and is internally pulled high. By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. PEN has no function during normal operation.

Resources

A comprehensive set of development tools, application notes, and datasheets are available for download on http://www.atmel.com/avr.

Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

About Code Examples

This datasheet contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

Capacitive touch sensing

The Atmel QTouch Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR microcontrollers. The QTouch Library includes support for the QTouch and QMatrix acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

The QTouch Library is FREE and downloadable from the Atmel website at the following location: www.atmel.com/qtouchlibrary. For implementation details and other information, refer to the Atmel QTouch Library User Guide - also available for download from the Atmel website.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	add two Registers	Rd ← Rd+ Rr	ZCNVSH	1
ADC	Rd, Rr	add with Carry two Registers	Rd ← Rd+ Rr +C	ZCNVSH	1
ADIW	Rdl,K	add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	ZCNVSH	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd- Rr	ZCNVSH	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd- K	ZCNVSH	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd- Rr - C	ZCNVSH	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd- K -C	ZCNVSH	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	ZCNVSH	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd• Rr	ZNV	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd• K	ZNV	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	ZNV	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	ZNV	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	ZNV	1
COM	Rd	One's Complement	Rd ← 0xFF − Rd	ZCNVS	1
NEG	Rd	Two's Complement	Rd ← 0x00 − Rd	ZCNVSH	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	ZNV	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd• (0xFF - K)	ZNV	1
INC	Rd	Increment	Rd ← Rd + 1	ZNV	1
DEC	Rd	Decrement	Rd ← Rd - 1	ZNV	1
TST	Rd	Test for Zero or Minus	Rd ← Rd• Rd	ZNV	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	ZNV	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	ZC	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	ZC	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	ZC	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	ZC	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	ZC	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	ZC	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd − Rr	Z,N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd − Rr − C	Z,N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd − K	Z,N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half-Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half-Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move between registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
ELPM		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3
ELPM	Rd, Z+	Extended Load Program Memory and Post-Inc	Rd ← (RAMPZ:Z), RAMPZ:Z ← RAMPZ:Z + 1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		For On-chip Debug Only		None	N/A

Instruction Set Summary (Continued)

Mnemonic	Operands	Description	Operation	Flags	#Clocks
SEV		Set Two's Complement Overflow	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Ordering Information

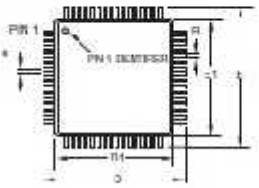
Speed (MHz)	Power Supply	Ordering Code[1]	Package[2]	Operation Range
8	2.7 – 5.5V	ATmega128L-8AU ATmega128L-8AUR[3] ATmega128L-8MU ATmega128L-8MUR[3]	64A 64A 64M1 64M1	Industrial (-40°C to 85°C)
16	4.5 – 5.5V	ATmega128-16AU ATmega128-16AUR[3] ATmega128-16MU ATmega128-16MUR[3]	64A 64A 64M1 64M1	
8	2.7 – 5.5V	ATmega128L-8AN ATmega128L-8ANR[3] ATmega128L-8MN ATmega128L-8MNR[3]	64A 64A 64M1 64M1	Extended (-40°C to 105°C)
16	4.5 – 5.5V	ATmega128-16AN ATmega128-16ANR[3] ATmega128-16MN ATmega128-16MNR[3]	64A 64A 64M1 64M1	

Notes: 1. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
2. The device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
3. Tape and Reel

Package Type	
64A	64 lead, 14 x 14 x 1.0mm, Thin Profile Plastic Quad Flat Package (TQFP)
64M1	64 pad, 9 x 9 x 1.0mm, Quad Flat No Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

64A



COMMON DIMENSIONS
(Unit of Measure = mm)

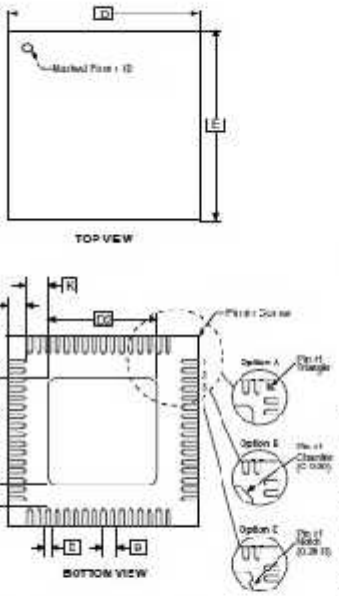
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	15.75	16.00	16.25	
D1	13.90	14.00	14.10	Note 2
E	15.75	16.00	16.25	
E1	13.90	14.00	14.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation AEB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

2010-10-20

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	64A, 64-lead, 14 x 14 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Plastic Quad Flat Package (TQFP)	64A	C

64M1



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	(1.00)	1.00	1.00	
A1		0.02	0.05	
b	0.18	0.23	0.30	
D	8.90	9.00	9.10	
D2	5.30	5.40	5.50	
E	8.90	9.00	9.10	
E2	5.30	5.40	5.50	
e		0.50 BSC		
L	0.25	0.40	0.45	
K	1.25	1.40	1.55	

Notes:
1. JEDEC Standard MO-220, (SAW Singulation) Fig. 1, VMMC.
2. Dimension and tolerance conform to ASME Y14.5M-1994.

2010-10-20

2325 Orchard Parkway San Jose, CA 95131	TITLE **64M1**, 64-pad, 9 x 9 x 1.0 mm Body, Lead Pitch 0.50 mm, 5.40 mm Exposed Pad, Micro Lead Frame Package (MLF)	DRAWING NO. 64M1	REV. H

LAMPIRAN B
(*DATA SHEET WATER FLOW*)

Specification

Mini. Wokring Voltage	DC 4.5V
Max. Working Current	15mA(DC 5V)
Working Voltage	5V~24V
Flow Rate Range	1~60L/min
Load Capacity	≤10mA(DC 5V)
Operating Temperature	≤80℃
Liquid Temperature	≤120℃
Operating Humidity	35%~90%RH
Water Pressure	≤2.0MPa
Storage Temperature	-25℃~+80℃
Storage Humidity	25%~95%RH

Mechanic Dimensions

Sensor Components

No.	Name	Quantity	Material	Note
1	Valve body	1	PA66+33%glass fiber	
2	Stainless steel bead	1	Stainless steel SUS304	
3	Axis	1	Stainless steel SUS304	
4	Impeller	1	PCM	
5	Ring magnet	1	Ferrite	
6	Middle ring	1	PA66+33%glass fiber	
7	O-seal ring	1	Rubber	
8	Stainless steel ring	1	Rubber	
9	Cover	1	PA66+33%glass fiber	
10	Screw	4	Stainless steel SUS304	
11	Cable	1	1007 28AWG	

Usage Example

Note: This example is abstracted from the forum, which was done by Charles Gantt. Thanks for his contribution.Let's see how it works.

Reading Water Flow rate with Water Flow Sensor

This is part of a project I have been working on and I thought I would share it here since there have been a few threads on how to read water flow rate in liters per hour using the Water Flow Sensor found in the Seeed Studio Depo. It uses a simple rotating wheel that pulses a hall effect sensor. By reading these pulses and implementing a little math, we can read the liquids flow rate accurate to within 3%. The threads are simple G3/4 so finding barbed ends will not be that hard.

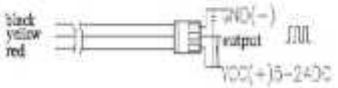
Hardware Installation

You will need Seeeduino / Arduino ,Water Flow Sensor,10K resistor,a breadboard and some jumper wires.

Wiring up the Water Flow Sensor is pretty simple. There are 3 wires: Black, Red, and Yellow. Black to the Seeeduino's ground pin Red to Seeeduino's 5v pin The yellow wire will need to be connected to a 10k pull up resistor and then to pin 2 on the Seeeduino.

Wiring Diagram

The external diameter of thread the connections use is 1.4mm.



Output Table

Pulse frequency (Hz) in Horizontal Test= 5.5Q. Q is flow rate in L/min. (Results in +/- 3% range)

Output pulse high level	Signal voltage >4.5 V(input DC 5 V)
Output pulse low level	Signal voltage <0.5V(input DC 5V)
Precision	3% (Flow rate from 1L/min to 10L/min)
Output signal duty cycle	40%~60%

LAMPIRAN C
(*LISTING PROGRAM* CODEVISION AVR)

```
/*****************************************************
**
This program was produced by the
CodeWizardAVR V2.05.3 Standard
Automatic Program Generator
© Copyright 1998-2011 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 30/06/2018
Author  :
Company :
Comments:


Chip type               : ATmega128
Program type            : Application
AVR Core Clock frequency: 16,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 1024
*****************************************************
*/

#include <mega128.h>
#include <stdio.h>
#include <delay.h>
#include <stdlib.h>
#include <alcd.h>


#define up      PINA.0
```

```
#define down    PINA.1
#define ok      PINA.2
#define valve   PORTA.3
#define mov_a1  PORTA.4
#define mov_a2  PORTA.5
#define mov_b1  PORTA.6
#define mov_b2  PORTA.7
#define b3 PORTE.2
#define b2 PORTE.3
#define b1 PORTE.4
#define b0 PORTE.5

// ---------------------------------------------- DEKLARASI SENSOR FLOW ---------------------------------------------------- //

float freq;
char buff0[30];
int dur,m=0;
int vol,step;

// ---------------------------------------------- DEKLARASI SENSOR PRESSURE ---------------------------------------------------- //

eeprom int sp;
int flag=0,simpan_sp;
float data_mpx;
float tekanan;
int x=43, y=255;
char buff1[30];
char temp[16];

// ---------------------------------------------- DEKLARASI SENSOR LEVEL ---------------------------------------------------- //

int adc,adc2, av_adc2,av_adc,data_av_adc2,data_av_adc, i;
char buff2[30];
```

```
char buff5[30];
char buff6[30];

// ---------------------------------------------- DEKLARASI RTC ----
---------------------------------------------------- //

unsigned char sc,mt,hr;
unsigned char mg,dd,mm,yy;
char buff3[30];
char buff4[30];

// ---------------------------------------------- BATAS SUCI
DEKLARASI ---------------------------------------------------- //


#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
```

```
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

#pragma used+
char getchar1(void)
{
char status,data;
while (1)
    {
    while (((status=UCSR1A) & RX_COMPLETE)==0);
    data=UDR1;
    if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
       return data;
    }
}
#pragma used-

// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
```

```
#define_ALTERNATE_PUTCHAR_
#include <stdio.h>
#define USART0 0              // agar pembacaan tidak acak
#define USART1 1
unsigned char poutput;

void putchar(char c)
{
switch (poutput)
{
case USART0: // the output will be directed to USART0
while ((UCSR0A & DATA_REGISTER_EMPTY)==0);
UDR0=c;
break;

case USART1: // the output will be directed to USART1
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
break;
};
}

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
// Place your code here
//   m++;
}

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
```

```
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}


//#include <spi.h>
// I2C Bus functions
//#include <i2c.h>

#include <twi.h>

// DS1307 Real Time Clock functions
#include <ds1307_twi.h>


// ---------------------------------------- KODING SENSOR + AKTUATOR FLOW ---------------------------------------- //

// SENSOR WATERFLOW //

void baca_flow(){
   TIMSK=0x04;
   TCCR1B=0x07;
   delay_ms(100);
   TCCR1B=0x00;
   TIMSK=0x00;
   dur=TCNT1;
   freq=(((dur+m*65536)*600)/5.4)*0.0166;
   vol=freq;
```

```
    TCNT1=0x0000;
    m=0;

}

// STEPPER //

 void searah_jj(){
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
 }
 void berlawanan_jj(){
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
  b0=1; b1=0; b2=0; b3=0;
  delay_ms(5);
```

```
  b0=0; b1=1; b2=0; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=1; b3=0;
  delay_ms(5);
  b0=0; b1=0; b2=0; b3=1;
  delay_ms(5);
}
void stopped(){
  b3=0; b2=0; b1=0; b0=0;
}
/*void stepper_buka100(){
    for(step=0;step<100;step++){
        berlawanan_jj();
        delay_ms(100);
    }
    stopped();
}
void stepper_buka20(){
    for(step=0;step<20;step++){
        berlawanan_jj();
        delay_ms(100);
    }
    stopped();
}
void stepper_tutup(){
    for(step=0;step<20;step++){
        searah_jj();
        delay_ms(100);
    }
    stopped();
}*/


// ------------------------------------------ KODING SENSOR + AKTUATOR PRESSURE ------------------------------------------
//
```

```
// SENSOR MPX //

void baca_tekanan(){
   data_mpx=read_adc(3);
   tekanan=((data_mpx-x)/(y-x))*7;
}

// ---------------------------------------- KODING SENSOR + AKTUATOR LEVEL ---------------------------------------- //

// SENSOR LEVEL SWITCH //

void baca_level_switch_atas(){
   for(i=0;i<20;i++){
       adc=read_adc(0);
       av_adc=av_adc+adc;
       delay_ms(10);
   }
   av_adc=av_adc/20;
   if(av_adc==0){
      data_av_adc=1;
   }
   else{
      data_av_adc=0;
   }
}
void baca_level_switch_bawah(){
   for(i=0;i<20;i++){
       adc2=read_adc(1);
       av_adc2=av_adc2+adc2;
       delay_ms(10);
   }
   av_adc2=av_adc2/20;
   if(av_adc2==0){
      data_av_adc2=1;
```

```
    }
    else{
       data_av_adc2=0;
    }
}

// ------------------------------------------------- TAMPIL LCD -------
------------------------------------------------- //

void tampil_lcd() {


// FLOW //
lcd_gotoxy(0,1);
lcd_putsf("F:");
ftoa(freq,2,buff0);
lcd_puts(buff0);

// PRESSURE //

lcd_gotoxy(0,2);
lcd_putsf("P:");
ftoa(tekanan,2,buff1);
lcd_puts(buff1);


// LEVEL //

lcd_gotoxy(0,0);
sprintf(buff2,"LSA:%d LSB:%d",data_av_adc,data_av_adc2);
lcd_puts(buff2);

sprintf(buff5,"%d" ,data_av_adc);
sprintf(buff6,"%d" ,data_av_adc2);

}
```

```
// ------------------------------------------------------ RTC ----------------
------------------------------------------------ //


void get_time()
{
   rtc_get_time(&hr,&mt,&sc);
   rtc_get_date(&mg,&dd,&mm,&yy);
}




void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T
State1=T State0=T
PORTA=0x0F;
DDRA=0b11111000;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In
Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T
State1=T State0=T
PORTB=0x00;
DDRB=0x00;
```

```
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0xFF;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
```

```
PORTG=0x00;
DDRG=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x05;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
```

```
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer3 Stopped
// Mode: Normal top=0xFFFF
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
```

```
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x04;
ETIMSK=0x00;

// USART0 initialization

UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization

UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;
```

```
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

twi_master_init(100);
rtc_init(0,0,0);

// Alphanumeric LCD initialization
// Connections are specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTC Bit 0
// RD - PORTC Bit 1
// EN - PORTC Bit 2
// D4 - PORTC Bit 4
// D5 - PORTC Bit 5
```

```
// D6 - PORTC Bit 6
// D7 - PORTC Bit 7
// Characters/line: 16
lcd_init(20);

// Global enable interrupts
#asm("sei")



//stepper_buka20();
valve=1;  //aktif high, buka
mov_a1=1;
mov_a2=0;
mov_b1=1;
mov_b2=0;
delay_ms(1000);
mov_a1=1;
mov_a2=1;
mov_b1=1;
mov_b2=1;

// SET WAKTU //
hr = 22;
mt = 12;
sc = 00;
dd = 18;
mm = 07;
yy = 18;

rtc_set_time(hr,mt,sc);
rtc_set_date(mg,dd,mm,yy);

while (1)
      {
      // Place your code here
```

```
        while(up&&down&&ok){
        baca_level_switch_atas();
        baca_level_switch_bawah();
        baca_tekanan();
        baca_flow();
        tampil_lcd();
        get_time();


        // DATA LOGGER //
        sprintf(buff4, "%02d:%02d:%02d/%02d:%02d:%02d ; %s
L/min\n\r",dd,mm,yy,hr,mt,sc,buff0);
        poutput = USART1;
        puts(buff4);


        // HMI INTEGRASI //
        sprintf(buff3,"%s L/min ; %s Bar ;LSA=%s;LSB=%s \r",
buff0,buff1,buff5,buff6);
        poutput = USART0;
        puts(buff3);


       if(vol>0&&vol<16){
           lcd_gotoxy(1,3);
           lcd_puts("mov tutup");
           //lcd_putsf("cw ");
           berlawanan_jj();


        }
        else if(vol>16.5){
           //lcd_gotoxy(12,0);
           //lcd_putsf("ccw ");
           searah_jj();
```

```
        lcd_gotoxy(1,3);
        lcd_putsf("mov buka");

    }
    else {
        stopped();
        lcd_gotoxy(1,3);
        lcd_putsf("mov henti");

    }



    if(data_av_adc==1 && data_av_adc2==1){
        valve=0;
    }


    if(tekanan>=simpan_sp){
        mov_a1=1;
        mov_a2=0;
        mov_b1=0;
        mov_b2=1;
    }

    else if(tekanan<simpan_sp){
        mov_a1=0;
        mov_a2=1;
        mov_b1=1;
        mov_b2=0;
    }

//  }


  if(!up){
```

```
      delay_ms(50);
      sp++;
      sprintf(temp,"%d ",sp);
      lcd_gotoxy(12,0);
      lcd_puts(temp);
    }
    else if(!down){
      delay_ms(50);
      sp--;
      sprintf(temp,"%d ",sp);
      lcd_gotoxy(12,0);
      lcd_puts(temp);
    }
    else if(!ok){
      delay_ms(50);
      simpan_sp=sp;
      flag=1;
      lcd_clear();
      delay_ms(1000);
      lcd_gotoxy(0,0);
      sprintf(temp,"%d",simpan_sp);
      lcd_puts(temp);
      delay_ms(1000);
      lcd_clear();
    }
      }
    }
}
```

LAMPIRAN D
(*LISTING PROGRAM* VISUAL BASIC)

```
Imports System
Imports System.IO
Imports System.IO.Ports
Imports System.Threading
Imports System.Collections.Generic

Public Class Form1
    Dim data_in As String

    Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load

    Private Sub serial_DataReceived(sender As Object, e As
IO.Ports.SerialDataReceivedEventArgs) Handles
serial.DataReceived
        data_in = serial.ReadLine

    End Sub

    Private Sub timer_Tick(sender As Object, e As EventArgs)
Handles timer.Tick
        Dim data_split As String() = data_in.Split(";")
        lbl_masukflow.Text = data_split(0)
        lbl_masukpressure.Text = data_split(1)
        lbl_masukLSA.Text = data_split(2)
        lbl_masukLSB.Text = data_split(3)

    End Sub

    Private Sub btn_start_Click(sender As Object, e As EventArgs)
Handles btn_start.Click
serial.PortName = "COM8"
        serial.BaudRate = "9600"
```

```
        Try
            serial.Open()
            If serial.IsOpen Then
                timer.Enabled = True
                MessageBox.Show("Success", "Note")
                    End If
        Catch ex As Exception
            MessageBox.Show("Failed", "Note")
        End Try
    End Sub

    Private Sub btn_stop_Click(sender As Object, e As EventArgs)
Handles btn_stop.Click
        timer.Enabled = False
        serial.Close()
    End Sub


End Class
```

BIODATA PENULIS

Penulis dilahirkan di Surabaya pada tangal 8 Juni 1997 dengan diberi nama Dyah Rengganis Permata Dewi. Penulis telah menyelesaikan pendidikan dasar selama 6 tahun di SDN Pacarkeling VII/188 Surabaya pada tahun 2009, SMP Negeri 37 Surabaya pada tahun 2012, SMA Negeri 15 Surabaya pada tahun 2015, dan pada tahun 2018 ini, penulis mampu menyelesaikan pendidikan Diploma di Program Studi D3 Teknologi Instrumentasi, Departemen Teknik Indtrumentasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Bagi pembaca yang memiliki kritik, saran ataupun ingin berdiskusi lebih lanjut mengenai tugas akhir ini, dapat menghubungi melalui alamat email dyahrengganis88@gmail.com.